PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA
MAESTRÍA EN REDES DE COMUNICACIONES

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE:

MÁSTER EN REDES DE COMUNICACIONES

TEMA:

"ESTUDIO COMPARATIVO MEDIANTE SIMULACIÓN DE LAS VARIANTES DE LA TECNOLOGIA OPC-UA Y SU UTILIZACIÓN EN CONTROLADORES INDUSTRIALES"

FERNANDO JAVIER JACOME SAGÑAY

Quito – 2016

DECLARACIÓN

Yo, Jácome Sagñay Fernando Javier portador de la cedula de ciudadanía No. 1712469566, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

_____

Fernando Javier Jácome Sagñay
CC # 1712469566

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Fernando Javier Jácome Sagñay, bajo mi supervisión.

_____

PHD. Gustavo Chafla

DIRECTOR DEL PROYECTO

AGRADECIMIENTO

Primeramente quiero agradecer a Dios por haberme permitido culminar este nuevo paso en mi vida profesional. Además, quiero agradecer a mi familia por darme soporte para culminar este objetivo y a todas las personas que me ayudaron para la elaboración y finalización de este proyecto.

Fernando Jácome

DEDICATORIA

Este proyecto está dedicado a Dios, a mi papá que esta con él, a mi esposa y mis hijos, a mi mamá y hermanos, y a todas aquellas personas que me ayudaron a culminar con éxito este gran paso en mi carrera profesional.

Fernando Jácome.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Este trabajo de investigación presenta un estudio comparativo de las alternativas del protocolo de comunicaciones OPC UA mediante simulación de la comunicación entre un cliente y servidor OPC UA y además, la simulación y uso de la comunicación entre un sistema SCADA y un controlador industrial basado en PC. Mediante la aplicación de diferentes escenarios reales de pruebas utilizando máquinas virtuales y aplicaciones OPC UA se realiza la captura de datos en cada una de las alternativas estudiadas para obtener una estimación del ancho de banda utilizado para la lectura de datos del tipo doble en varias cantidades específicas de datos y con muestreo de datos de 1 segundo hasta 100 milisegundos. Por último se realiza una simulación de un sistema SCADA y un controlador industrial basado en PC y se procede a obtener la captura de datos para estimar el comportamiento y el uso del ancho de banda.

Esto permite estimar el ancho de banda utilizado por el protocolo OPC UA aplicando las configuraciones que más se apeguen a las necesidades como pueden ser aplicación o no de directivas de seguridad, cifrado con o sin firma digital entre otras

Con los resultados obtenidos de ésta investigación se pretende establecer el ancho de banda utilizado por las aplicaciones OPC UA para un determinado tipo de dato y realizar un posible remplazo de la anterior tecnología OPC utilizada en comunicaciones SCADA la cual no es independiente del sistema operativo y no posee las características de seguridad actuales.

1 CAPÍTULO I: INTRODUCCIÓN AL PROYECTO DE INVESTIGACIÓN.

1.1 INTRODUCCIÓN

El presente trabajo se enfoca en el estudio, simulación e implementación de la tecnología OPC UA utilizada en comunicaciones de interfaces hombre-máquina con controladores industriales basados en PC. El principal motivo de este proyecto de tesis es el uso de la nueva tecnología OPC UA como remplazo de la tecnología OPC DA, la cual presenta algunos inconvenientes.

Primeramente se realizará un estudio del estado del arte de la tecnología OPC UA y sus principales especificaciones. Posteriormente se presentará un estudio de las principales aplicaciones comerciales que se tiene al momento para implementar la nueva tecnología OPC UA. Finalmente se realizará una simulación de una interfaz hombre- máquina con comunicación OPC UA funcionando en una PC y controladores industriales. La interface deberá ser hecha en una de las aplicaciones interfaz hombre-máquina para mostrar datos de estados digitales y monitoreo analógico que será simulado y enviado por los controladores industriales. Los controladores industriales deberán tener la posibilidad de comunicación OPC UA. Este trabajo persigue implementar la nueva tecnología OPC UA y realizar una simulación para comprobar su funcionamiento y demostrar el posible remplazo de tecnologías anteriores.

1.2 JUSTIFICACIÓN.

En muchas industrias en el Ecuador se tiene implementado varias tecnologías de comunicación entre Sistemas SCADA y controladores industriales y cada área de una planta industrial es un ambiente independiente uno de otro y por consiguiente con diferentes protocolos de comunicaciones, con lo cual resulta un problema transferir datos entre controladores de diferentes vendedores, entre diferentes sistemas SCADA y entre diferentes áreas de producción. Además de la interoperabilidad, en ambientes industriales

se necesita de seguridad en las comunicaciones que eviten posibles ataques que pueden realizar acciones dañinas sobre las máquinas del proceso. La tecnología OPC UA permite la estandarización e interconexión entre diferentes vendedores industriales y comunicaciones con seguridad, por consiguiente es un prospecto como la solución para este tipo de problema.

OPC UA provee una comunicación de red en forma segura, durable y estandarizada. Posee diferentes protocolos de comunicación, formas de cifrado y supervisión de tiempos de espera en interrupciones de red. La tecnología OPC UA tiene independencia de la plataforma y puede ser implementado en varias plataformas de hardware y sistemas operativos, permitiendo la conectividad entre varios productos de varios vendedores, para de esta forma facilitar la transmisión de datos entre controladores industriales, desarrolladores de software industrial y usuarios finales. OPC UA busca ser un estándar de intercambio de información de tiempo real con aceptación global con una comunicación rápida y segura. Puede ser utilizado en hardware de PCs, PLCs, micro-controllers y servidores de nube. Utilizable en sistemas operativos como Microsoft Windows, Android, Linux y Apple OSX. OPC UA permite la interoperabilidad desde el nivel de los sensores hasta el nivel de manejo de datos de la empresa.

OPC UA es fácil de configurar a través de firewalls y provee de varios mecanismos de seguridad. Además, posee dos tipos de métodos de comunicaciones, entre ellos el protocolo ultra rápido OPC binario y para la comunicación hacia Internet posee Servicios Web con SOAP/HTTP. Además, cada mensaje es transmitido de forma segura mediante el cifrado a 128 o 256 bits. De igual forma los mensajes son recibidos exactamente como fueron enviados debido a que son mensajes digitalmente firmados. También, cada aplicación, tanto clientes como servidores se deben identificar mediante certificados Open SSL para permitir cuales aplicaciones pueden conectarse. El protocolo Binario UA es el preferido para la comunicación entre sistemas de control y sistemas SCADA/HMI debido a su rapidez y a su fácil implementación entre dispositivos de bajos y medios recursos. Para comunicaciones con aplicaciones empresariales y sistemas de alto nivel, se utiliza Servicios Web que corren sobre HTTP.

La importancia de este proyecto busca la utilización de nuevas tecnologías para facilitar las comunicaciones industriales y la integración mediante un estándar de comunicaciones el cual resuelve muchos problemas de las tecnologías anteriores y ayuda a minimizar los costos de implementación y de migración.

Este estudio por consiguiente plantea investigar las capacidades de la infraestructura de comunicaciones de la tecnología OPC UA y realizar la implementación mediante una simulación de la comunicación mediante OPC UA entre un sistema SCADA y un controlador industrial.

1.3 ANTECEDENTES.

OPC UA tiene sus inicios en el año 2008 cuando fue lanzada y presentada como una arquitectura orientada al servicio e independiente de la plataforma que integra todas las funcionalidades de la especificación de OPC Clásica dentro de un marco de referencia extensible. OPC UA tiene entre sus principales objetivos tener una equivalencia funcional en cuanto a la anterior especificación OPC Clásica y poder mapear las funcionalidades anteriores hacia OPC UA por medio de Wrappers y Proxis. Otro de los objetivos que busca es la independencia de la plataforma y poder ser implementada tanto en servidores Microsoft hasta sistemas embebidos con sistemas operativos con un mínimo impacto. También busca ser más seguro mediante la implementación de cifrado, autenticación y auditoria (Protocolos AAA). Como se mencionó anteriormente busca ser extensible y poder añadir nuevas funcionalidades a las existentes. Por último, trata de implementar un modelado de la información de una forma comprensiva para definir información compleja (OPC-Foundation, 2015).

OPC UA es una arquitectura basada en cliente-servidor, no es únicamente una tecnología de comunicación entre controladores y sistemas de supervisión sino para comunicaciones desde el nivel de producción hasta el nivel de manejo de negocio (Schwarz & Borcsok, 2013), inclusive es considerado como una de las especificaciones para la aplicación de la Smart Grid en cuanto a modelado de la información y comunicaciones (Rohjans & Klaus, 2011).

Además, el estándar IEC 62541 es el estándar internacional en el cual fue publicado OPC UA como el estándar internacional para la comunicación vertical y horizontal en manufactura y automatización proveyendo interoperabilidad entre sistemas conectados. También, provee una base para la conectividad del Internet de las cosas (IoT) y de la iniciativa de Industria 4.0 (IEC TR 62541-1:2010 OPC Unified Architecture).

En lo que refiere a los productos de software desarrollados para la implementación de la tecnología OPC UA existen los productos certificados. Para que un producto sea certificado este debe pasar por un programa de certificación que asegura que el producto cumpla los requerimientos de OPC Foundation para interoperabilidad, confiabilidad y funcionamiento mínimo, y así minimizar los costos de integración de sistemas sabiendo que se tienen productos certificados (OPC-Foundation, 2015). Existe una lista para los productos certificados por OPC UA, entre los que están programas HMI, librerías, paquetes SDK (Software Developer Kit), Wrappers y Proxies.

Actualmente los países que están implementando fuertemente la tecnología OPC UA son los países de Alemania, Estados Unidos, Arabia Saudita entre otros, y se está utilizando en los campos de las empresas de transporte, empresas automotrices, manufactura química, monitoreo de energía, empresas de alimentos y bebidas, gas y petróleo, empresas de fundición de metales y plantas de tratamiento de aguas.

1.4 OBJETIVOS

1.4.1 Objetivo General

Realizar el estudio de la tecnología OPC UA para la comunicación e intercambio de información remota entre controladores industriales. Comparar las diferentes alternativas que se encuentran disponibles para el uso de la tecnología OPC UA y realizar la simulación para la comparación entre las alternativas actuales (sin seguridades, cifrado y firma digital).

1.4.2 Objetivos Específicos

- Estudiar el estado del arte de la tecnología OPC-UA.

- Explorar las capacidades de la tecnología OPC-UA para monitoreo remoto.
- Realizar una comparación entre las alternativas actuales que se encuentran disponibles para la implementación de la tecnología OPC UA.
- Simular las alternativas de la tecnología OPC UA y obtener datos para la comparación de estas alternativas.
- Realizar un estudio del tráfico de datos que se intercambian bajo OPC UA, y de esta forma conocer el ancho de banda que se consume para el correcto dimensionamiento de enlaces.
- Investigar el uso de sistemas SCADA con tecnología OPC UA.
- Realizar la simulación de sistemas SCADA con tecnología OPC UA.

1.5 RESUMEN DE CONTENIDO DE CAPÍTULOS.

Se inicia el presente proyecto con el primer capítulo en donde se realiza una breve introducción indicando cuales son los antecedentes y razones para el desarrollo del presente proyecto.

En el segundo capítulo se realiza una descripción del estado del arte de la tecnología OPC UA.

En el tercer capítulo se realiza un análisis comparativo de las diferentes alternativas de la tecnología OPC UA mediante simulación y conocer el ancho de banda utilizado por la tecnología.

En el cuarto capítulo se desarrolla la implementación y configuración del software de simulación SCADA y su comunicación con el controlador industrial basado en PC.

En el quinto capítulo se describirán las conclusiones y recomendaciones del presente proyecto.

2 CAPÍTULO 2: ESTADO DEL ARTE.

2.1 INTRODUCCIÓN A LA TECNOLOGÍA OPC UA.

Cuando se habla de una tecnología nueva, uno se encuentra con dos objetivos a tratar de obtener, el primero es cumplir con el rendimiento y la confiabilidad, y el segundo es cumplir con la seguridad, estos objetivos son cumplidos en la tecnología OPC UA. Una de las interrogantes es cuál es la mejor alternativa para desarrollar un estándar, esto debido a que un sistema para desarrollar la tecnología OPC UA puede ser remplazado por otro sistema más actual y mejor, inclusive un sistema de desarrollo puede ser dado de baja si se realiza una actualización de un sistema operativo antiguo a uno moderno. Por la razón presentada anteriormente, para ser un estándar abierto a futuras tecnologías el grupo de trabajo de OPC UA ha publicado servicios y conceptos de una forma abstracta y de tal forma, especificar asignaciones de tecnologías para su implementación. Las asignaciones de la tecnología OPC UA para el intercambio de la información se basan en tres aspectos: codificación de la información, seguridad en la comunicación y el transporte de los datos. (Mahnke, Leitner, & Damm, 2009). La figura 2.1 muestra las capas funcionales utilizadas en OPC UA.

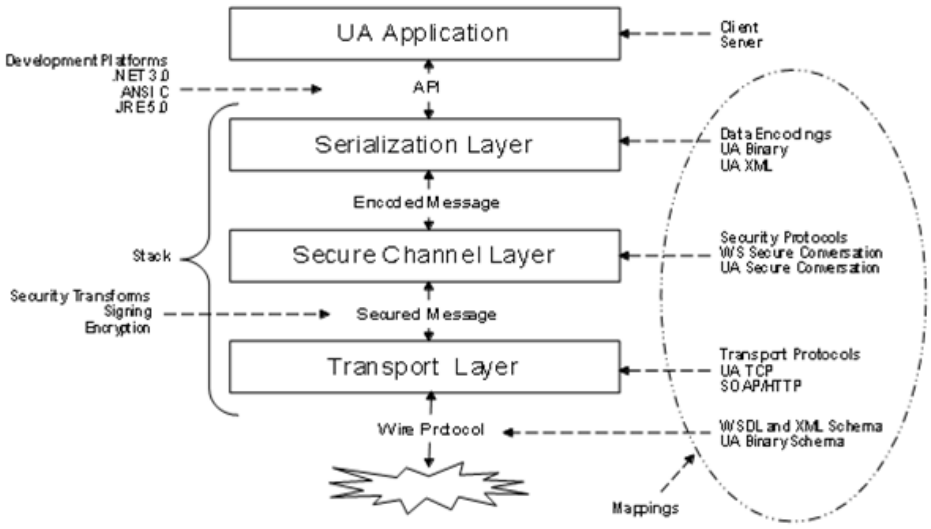

Figura 2.1: Capas del stack de la tecnología OPC UA. Adaptado: OPC Unified Architecture, Specification, Part 6.

En la figura 2.1 se puede ver las capas responsables de la codificación, seguridad y transporte que forman el llamado stack. El stack son componentes de software que pueden ser utilizados por otras aplicaciones para la implementación del estándar. Para cada capa del stack se han definido dos opciones de implementación y la posibilidad de adicionar más opciones en futuro. Para que un producto OPC UA pueda ser desarrollado tiene que cumplir con al menos una de las opciones de cada capa del stack. (Mahnke, Leitner, & Damm, 2009).

2.2 FUNDAMENTOS DE LA TECNOLOGÍA OPC UA.

2.2.1 Codificación de Datos.

La codificación de datos trata la serialización de los mensajes de los servicios implementados en el estándar como son los parámetros de entrada y de salida en un formato para envió hacia la red, generalmente en una secuencia de bytes. OPC UA define dos tipos de codificación: Binario OPC UA y XML. Para ambos tipos, la especificación OPC UA define la codificación para tipos de datos primitivos hasta datos más complejos y estructurados. En la Tabla 2.1 se muestra los tipos de datos incorporados en el estándar.

2.2.1.1 *Codificación Binaria OPC UA.*

La codificación binaria OPC UA trata de la representación de datos simples hasta complejos en una secuencia de bits que pueden tener el estado de 0 o 1 lógico. Debido a que este tipo de cifrado esta en bits, la serialización y deserialización es más fácil y rápida siendo esta es la más eficiente y utilizada por sistemas que tienen bajos recursos o que necesitan de rapidez.

Tabla 2.1 : Tipos de datos incorporados en el estándar OPC UA. Adaptado: OPC Unified Architecture, Specification, Part 6.

ID	Nombre	Descripción
1	Boolean	Un valor lógico de dos estados (verdadero o falso).
2	SByte	Un valor entero entre −128 y 127.
3	Byte	Un valor entero entre 0 y 255.
4	Int16	Un valor entero entre −32 768 y 32 767.
5	UInt16	Un valor entero entre 0 y 65 535.
6	Int32	Un valor entero entre −2 147 483 648 y 2 147 483 647.
7	UInt32	Un valor entero entre 0 y 4 294 967 295.
8	Int64	Un valor entero entre −9 223 372 036 854 775 808 y 9 223 372 036 854 775 807
9	UInt64	Un valor entero entre 0 y 18 446 744 073 709 551 615.
10	Float	Un valor de punto flotante de precisión simple formato IEEE (32 bit).
11	Double	Un valor de punto flotante de precisión doble formato IEEE (64 bit).
12	String	Una secuencia de caracteres Unicode.
13	DateTime	Una instancia en tiempo.
14	Guid	Un valor de 16 bytes que puede ser usado como un identificador global único.
15	ByteString	Una secuencia de octetos.
16	XmlElement	Un elemento XML.
17	NodeId	Un identificador para un nodo en el espacio de direcciones de un servidor OPC UA.
18	ExpandedNodeId	Un NodeId que permite especificar un espacio de nombres URI en lugar de un indice.
19	StatusCode	Un identificador numérico para un error o condición que es asociado con un valor o una operación.
20	QualifiedName	A nombre calificado por un espacio de nombres.
21	LocalizedText	Texto entendible con un identificador local opcional.
22	ExtensionObject	Una estructura que contiene una tipo de dato específico de una aplicación que no puede ser reconocido por el receptor.
23	DataValue	Un valor de un dato con un codigo de estado asociado y una estampilla de tiempo.
24	Variant	Una unión de todos los tipos descritos anteriormente.
25	DiagnosticInfo	Una estructura que contiene información detallada de error y diagnóstico asociado con un código de estado.

En la siguiente figura 2.2 se muestra un ejemplo de la codificación binaria de un tipo de dato entero. Este ejemplo muestra el valor de 2000000000 (hexadecimal=77359400) codificado como 32 bits o 4 bytes.

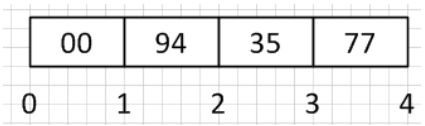

Figura 2.2: Codificación de un tipo de dato Entero de 32 bits. Fuente: Autor.

En el caso de tipos de datos complejos se tiene una estructura conformada por diferentes tipos de datos primitivos. Para el caso de enumeraciones estas son codificadas como valores de tipo Int32. En el caso de arreglo de datos se codifican como una secuencia de datos que tienen un elemento inicial que indica el número total de elementos, este campo de longitud esta codificado como un valor de tipo Int32. Para el caso de estructuras se tiene una secuencia de campos que indican el tipo de dato, la longitud en bytes de los datos y los datos propiamente dichos.

En la figura 2.3 se muestra una instancia de un tipo de dato complejo llamado Type1 codificado en un tipo de dato ExtensionObject. El tipo de dato ExtensionObject posee por defecto los campos de TypeId, Encoding y Length. En los siguientes campos se observa una estructura llamada Type1 formada por 4 datos: X, NoOfY, Y que es un arreglo formado por dos (2) elementos y Z. Cada elemento del arreglo Y es una estructura llamada Type2 formada por dos (2) datos los cuales son A y B. Todos los elementos están formados por 4 bytes a excepción del campo Encoding de longitud de 1 byte y son del tipo Int32.

Tabla 2.2: Ejemplo de la codificación de una estructura en OPC UA. Adaptado: OPC Unified Architecture, Specification, Part 6.

Campo	Bytes	Valor
Type Id	4	El identificador para Type1
Encoding	1	0x1 para ByteString
Length	4	28
X	4	El valor del campo 'X'
NoOfY	4	2
Y.A	4	El valor del campo'Y[0].A'
Y.B	4	El valor del campo'Y[0].B'
Y.A	4	El valor del campo 'Y[1].A'
Y.B	4	El valor del campo'Y[1].B'
Z	4	El valor del campo 'Z'

2.2.1.2 *Codificación XML.*

XML es un estándar en lo que se refiere a formato de datos que pueden ser entendidos y utilizados por cualquier aplicación que tenga un analizador XML. Algunas de las aplicaciones que pueden consumir datos con el formato XML son los sistemas MES (Manufacturing Executios System) que pueden intercambiar información a nivel empresarial como en los sistemas ERP (Enterprise Resource Planning). Debido a esto se espera que el estándar OPC UA tenga implementado la codificación XML.

Muchos de los tipos de datos de OPC UA incorporados con codificación XML usan el formato que se muestra en la parte 2 del esquema XML y publicado en la especificación XML [W3C04a y W3C04b]. Todos los tipos de datos incorporados en la especificación OPC UA pueden ser codificados en XML.

El prefijo xs: es utilizado como símbolo en el esquema XML. En la tabla 2.3 se muestra la sintaxis en XML de los diferentes tipos de datos entero. En la figura 2.3 se muestra un ejemplo de la sintaxis XML para un tipo de datos String. Por ultimo en la figura 2.4 se muestra la codificación en XML de un tipo de dato complejo o estructurado.

Tabla 2.3: Asignación de tipos de datos enteros codificados en XML. Adaptado: OPC Unified Architecture, Specification, Part 6.

Name	XML Type
SByte	xs:byte
Byte	xs:unsignedByte
Int16	xs:short
UInt16	xs:unsignedShort
Int32	xs:int
UInt32	xs:unsignedInt
Int64	xs:long
UInt64	xs:unsignedLong

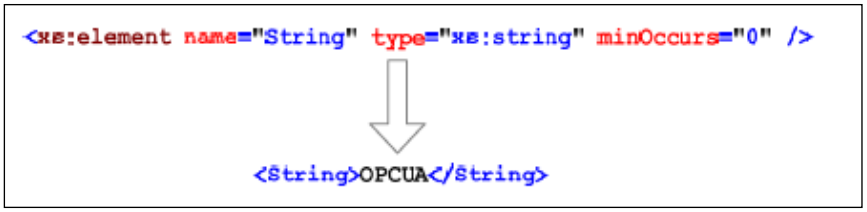

Figura 2.3: Ejemplo de la codificación XML de un tipo de dato String. Adaptado: OPC Unified Architecture. Wolfgang Mahnke. 2009.

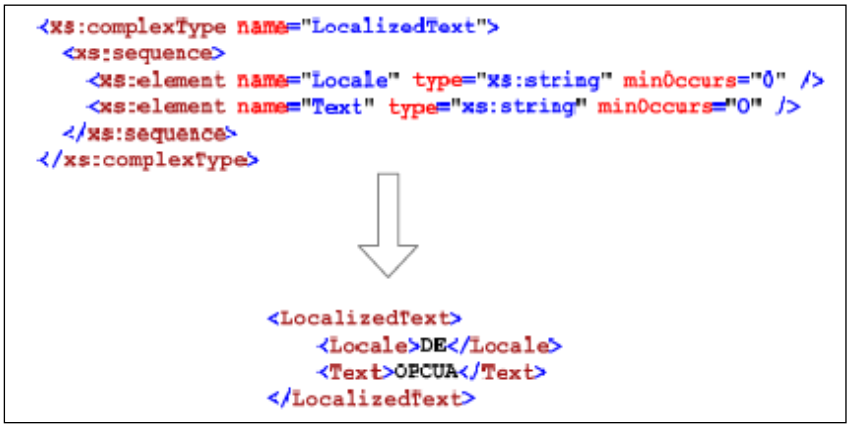

Figura 2.4: Ejemplo de la codificación XML de un tipo de dato complejo llamado LocalizedText. Adaptado: OPC Unified Architecture. Wolfgang Mahnke. 2009.

2.2.2 Protocolos de seguridad.

Todas las aplicaciones para poder comunicarse con la tecnología OPC UA debe crear un canal seguro (SecureChannel) y después establecer una sesión (Session), antes de consumir cualquier servicio implementado en la tecnología OPC UA, esto es conocido como una Conversación Segura (**Security Handshake**). Los protocolos de seguridad pueden soportar tres modos de seguridad: Ninguno (None), Firmado (Sign), y Firmado y cifrado (SignAndEncrypt). Si el modo de seguridad utilizado es Ninguno, no se usa ningún tipo de seguridad y el establecimiento de la conversación segura no es cifrado. En la figura 2.5 se muestra un diagrama del establecimiento de una conversación segura.

El set de algoritmos de seguridad utilizados durante una conversación segura es llamado la política de seguridad (Security Policies). El stack OPC UA debe estar implementado con el soporte de las políticas de seguridad expresadas en la especificación actual. Las aplicaciones deben especificar la política de seguridad mediante un URI (Uniform Resource Identifier), el cual es un conjunto de caracteres que identifica un recurso en red) que es comunicado al stack. En la tabla 2.4 se muestran las políticas de seguridad en OPC UA con su respectivo URI.

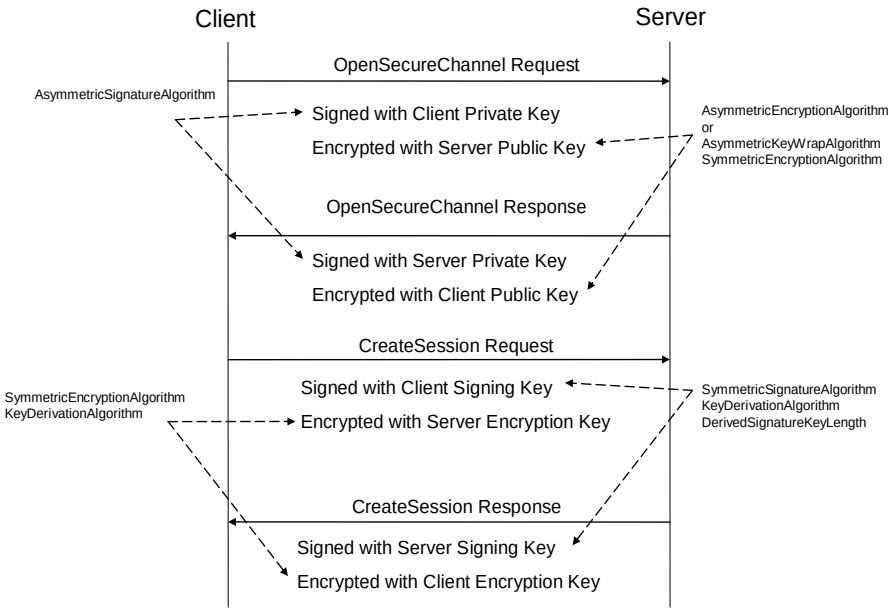

Figura 2.5: Security Handshake en OPC UA. Adaptado: OPC Unified Architecture, Specification, Part 6.

Tabla 2.4: Políticas de seguridad y su URI implementado en OPC UA. Adaptado: OPC Unified Architecture, Specification, Part 6.

Nombre	**Descripción**
PolicyUri	El URI asignado a la política de seguridad (*SecurityPolicy)*.
SymmetricSignatureAlgorithm	El URI para usar el algoritmo de firma simétrica.
SymmetricEncryptionAlgorithm	El URI para usar el algoritmo de cifrado simétrica.
AsymmetricSignatureAlgorithm	El URI para usar el algoritmo de firma asimétrica.
AsymmetricKeyWrapAlgorithm	El URI para usar el algoritmo de envoltura de llave asimétrica.
AsymmetricEncryptionAlgorithm	El URI para usar el algoritmo de cifrado asimétrica.
MinAsymmetricKeyLength	La longitud mínima para la llave asimétrica.
MaxAsymmetricKeyLength	La longitud máxima para la llave asimétrica.
KeyDerivationAlgorithm	El URI Para usar el algoritmo de derivación de llave.
DerivedSignatureKeyLength	La longitud en bits de la llave derivada usada para autenticación de mensajes.

Existen dos tipos de protocolos de seguridad en OPC UA, estos son, la conversación segura OPC UA (UA-SecureConversation) y la conversación Segura Web Services WS (WS-SecureConversation) y ambas son basadas en el establecimiento de conexión mediante certificados.

2.2.2.1 *Conversación segura UA.*

La conversación segura OPC UA (UASC) es la versión binaria de la conversación segura WS (Web Services) y provee de una conversación segura que no utiliza ni SOAP o XML.

UASC puede utilizar varios protocolos de transporte (TransportProtocols) que pueden tener un buffer con tamaños limitados. Debido a que puede haber protocolos de transporte con buffer limitados se deben romper los mensajes OPC UA en pedazos llamados *MessageChunks*, estos pedazos son pequeños en tamaño. En la figura 2.6 se muestran los protocolos de transporte (UA TCP y SOAP), los protocolos de seguridad (Conversación Segura UA y Conversación Segura WS) y la codificación con relación a la capa de transporte TCP y los puertos utilizados por cada protocolo. El buffer mínimo que requiere la conversación segura OPC UA es de 8196 bytes. Todas las seguridades son aplicadas a cada pedazo y no al mensaje entero. El stack debe tener implementado la aplicación de seguridades a cada MessageChunk y la habilidad de recibir los pedazos y ensamblar al mensaje completo.

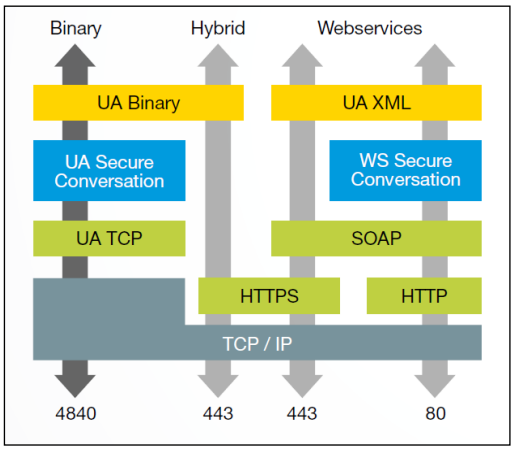

Figura 2.6: Protocolos de transporte, protocolos de seguridad y cifrado en OPC UA. Adaptado: OPC Unified Architecture, The interoperability Standard, OPC Foundation Brochure, 2013.

En la figura 2.7 se muestra la estructura del MessageChunk de una conversación segura OPC UA, en donde se puede observar las diferentes cabeceras de la que está formado y como las seguridades son implementadas a cada parte del mensaje.

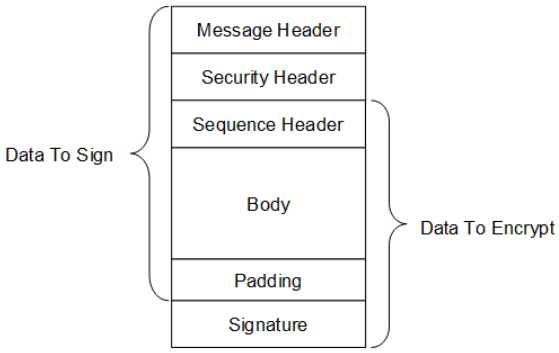

Figura 2.7: Estructura de un *MessageChunk* en una conversación segura OPC UA. Adaptado: OPC Unified Architecture, Specification, Part 6.

La tabla 2.5 muestra los campos de la cabecera de Mensaje (Message Header), el tipo de dato y su longitud.

Tabla 2.5: Estructura de la cabecera de Mensaje de un Message Chunk en UASC. Adaptado: OPC Unified Architecture, Specification, Part 6.

Nombre	Tipo de Dato	Descripción
MessageType	Byte[3]	Un codigo ASCII de 3 bytes que indica el tipo de mensaje.
		Los siguientes valores son definidos:
		MSG Un mensaje seguro con las llaves asociadas con el canal.
		OPN Un mensaje del servicio **OpenSecureChannel**.
		CLO Un mensaje del servicio **CloseSecureChannel**.
IsFinal	Byte	Un codigo ASCII de un byte que indica si el MessageChunk es el pedazo final del mensaje.
		Los siguientes valores son definidos:
		C Un pedazo intermedio.
		F El pedazo es el final.
		A El pedazo final (usado cuando ocurre un error y el mensaje es desechado).
MessageSize	UInt32	La longitud del MessageChunk, en bytes. Este valor incluye el tamaño de la cabecera del Mensaje.
SecureChannelId	UInt32	Un identificador unico asignado al canal seguro asignado por el servidor.
		Si el servidor recibe un Id de canal seguro que no reconoce, se devolvera un error a nivel de la capa de transporte.
		Cuando el servidor inicia el primer Id de canal seguro este debe ser unico despues de cada reinicio de comunicacion. Esto asegura que cuando hay un reinicio en el servidor no cause el uso de Ids anteriores que han sido asignados a clientes.

En la siguiente figura 2.8 se muestra la cabecera de mensaje en una conversación segura OPC UA sin seguridad captada por medio del programa Wireshark 2015.

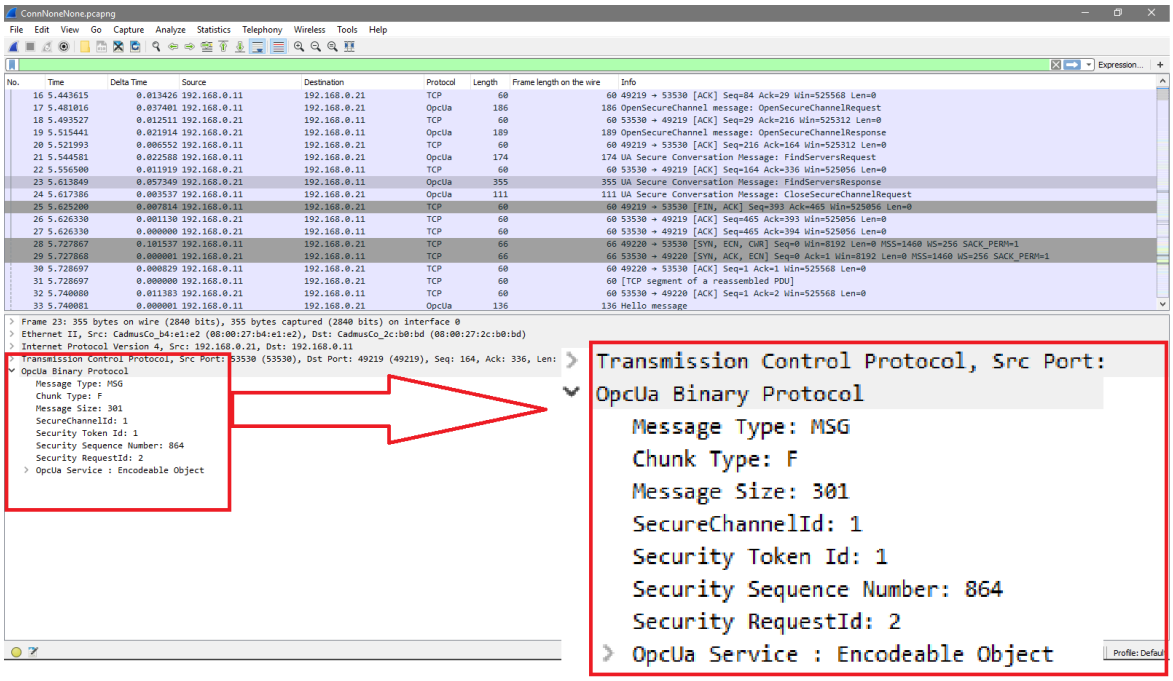

Figura 2.8: **Cabecera de mensaje UASC obtenida con el programa Wireshark**. Fuente: Autor.

La siguiente cabecera es la de seguridad, la cual especifica las opciones de criptografía que se aplicarán al mensaje. Hay dos versiones de la cabera de seguridad, una es para especificar algoritmos asimétricos y la segunda versión para algoritmos simétricos.

La versión de la cabecera de seguridad con algoritmos asimétricos se utiliza para el servicio de OpenSecureChannel. La tabla 2.6 muestra los campos de la cabecera de seguridad de la versión con algoritmos asimétricos.

Tabla 2.6: Cabecera de seguridad para algoritmos asimétricos. Adaptado: OPC Unified Architecture, Specification, Part 6.

Nombre	Tipo de dato	Descripción
SecurityPolicyUriLength	Int32	La longitud del URI de la política de seguridad en bytes. Este valor no debe exceder los 255 bytes.
SecurityPolicyUri	Byte[*]	El URI de la política de seguridad usada para asegurar el mensaje. Este campo esta codificado como UTF8 string sin un terminador null.
SenderCertificateLength	Int32	La longitud del certificado del trasmisor en bytes in bytes. Este valor no debe exceder el cálculo del máximo del tamaño del certificado en bytes.
SenderCertificate	Byte[*]	El certificado X509v3 asignado a la instancia de la aplicación que está enviando. Este es un codificado DER (Distinguished Encoding Rules) blob (Binary large object). La estructura de un certificado X509 es definido in X509. El formato DER para un certificado es definido en X690 Este indicará que llave privada fue usada para firmar el MessageChunk. El stack cerrará el canal y reportará un error a la aplicación si el certificado del trasmisor es demasiado largo para el tamaño del buffer que es soportado por la capa de transporte. Este campo sera null si el mensaje no está firmado. Si el certificado está firmado por el CA (Certification authority) el certificado del CA codificado en DER debe ser anexado después del certificado en el arreglo de bytes. Si el certificado del CA es también firmado por otro CA el proceso es repetido hasta que la cadena entera de certificados este en el buffer o si el valor calculado limitante MaxSenderCertificateSize sea alcanzado. (El proceso se detiene si el último certificado entero puede ser añadido sin exceder al límite MaxSenderCertificateSize). El receptor puede extraer los certificados del arreglo de bytes usando el tamaño del certificado contenido en la cabecera DER. El receptor que no maneja cadenas de certificados ignorará los bytes extras.
ReceiverCertificateThumbprintLength	Int32	La longitud del ReceiverCertificateThumbprint en bytes. La longitud de este campo es siempre de 20 bytes.
ReceiverCertificateThumbprint	Byte[*]	El thumbprint del certificado X509v3 asignado a la instancia de la aplicación del receptor. El thumbprint es SHA1 de la forma codificada DER del certificado. Este campo indica que llave pública fue usada para encriptar el MessageChunk. Este campo debe ser null si el mansaje no está encriptado.

El certificado del remitente debe ser de tamaño suficiente para estar incluido en un MessageChunk y tener por lo menos 1 byte en el cuerpo del envió. El tamaño del MessageChunk depende del protocolo de transporte pero debe ser mínimo de 8196 bytes. El tamaño de la firma asimétrica (AsymetricSignatureSize), si es utilizado, depende de los bits utilizados para la llave pública utilizada en el certificado del remitente.

La ecuación 1 indica como calcular el límite del tamaño del certificado del transmisor (MaxSendCertSize) para el cual debe tener en cuenta el tamaño de 12 bytes de la cabecera del mensaje, 4 bytes del campo SecurityPolicyUriLength, el campo

SecurityPolicyUri (SPU) el cual es un string UTF-8, 4 bytes del campo SenderCertificateLength de la cabecera de seguridad, 4 bytes del campo ReceiverCertificateThumbprintLength de la cabecera de seguridad, 20 bytes de la huella digital del certificado, 8 bytes del tamaño de la cabecera de secuencia, 1 byte mínimo como longitud del cuerpo de envio (SendBodyMsg), 1 byte para el campo Paddingsize si se aplica Padding, tamaño del Padding si se aplica, tamaño del Extrapadding y por último el tamaño de la firma asimétrica.

$$\mathbf{MaxSendCertSize} = \mathbf{MessageChunkSize} - \mathbf{16} - \mathbf{SecurityPolicyUri} - \mathbf{36} - \mathbf{SendBodyMsg} - \mathbf{1} - \mathbf{Padding} - \mathbf{ExtraPadding} - \mathbf{AsymmetricSignatureSize}. \quad \textbf{(2-1)}$$

La cabecera de seguridad que tiene algoritmos simétricos es usada para los mensajes que sean diferentes a los utilizados para establecer el canal seguro. La tabla 2.7 muestra la cabecera de seguridad cuando se usa algoritmos simétricos.

Tabla 2.7: Cabecera de seguridad para algoritmos simétricos. Adaptado: OPC Unified Architecture, Specification, Part 6.

Nombre	Tipo de dato	Descripción
TokenId	UInt32	Un identificador único para el SecurityToken del canal seguro utilizado para asegurar al mensaje. Este identificador es regresado al servidor en el mensaje de respuesta del servicio OpenSecureChannel. Si el servidor recibe un TokenId el cual no es reconocido, este responderá con un error a nivel de capa de transporte.

La cabecera de seguridad es seguida de la cabecera de secuencia, la cual indica el número de cada pedazo del mensaje. La Tabla 2.8 muestra los campos de la cabecera de secuencia.

Tabla 2.8: Cabecera de secuencia. Adaptado: OPC Unified Architecture, Specification, Part 6.

Nombre	Tipo de dato	Descripción
SequenceNumber	UInt32	Un número de secuencia incrementado monótonamente asignado por el remitente de cada MessageChunk enviado sobre el canal seguro.
RequestId	UInt32	Un identificador asignado por el cliente al mensaje de petición de comunicación OPC UA. Todos los MessageChunks de la petición y la respuesta asociada usan el mismo identificador.

Un número de secuencia (campo SecuenceNumber) nunca debe ser reusado por un TokenId. El token de seguridad tiene un tiempo de vida que debe ser suficiente para asegurar que nunca vuelva a usar el número de secuencia, si esto pasa el receptor envía un error a nivel de capa de transporte.

El número de secuencia debe ser incrementado para todos los mensajes y no debe reiniciarse a menos que sea mayor que 4 294 966 271 (máximo valor de un Int32 – 1024), el primer número antes de reiniciarse debe ser menor de 1024. El número de secuencia debe incrementarse en uno con cada pedazo del mensajea menos que el canal de comunicación sea interrumpido y reestablecido.

A continuación de la cabecera de seguridad se tiene el cuerpo propiamente del mensaje que esta codificado en OPC UA Binario, el cual es dividido en múltiples MessageChunks o pedazos. Cada pedazo del mensaje tiene una cola de seguridad (Security footer), la cual se muestra en la Tabla 2.9 a continuación.

Tabla 2.9: Cola de mensaje de UASC. Adaptado: OPC Unified Architecture, Specification, Part6.

Nombre	Tipo de dato	Descripción
PaddingSize	Byte	El tamaño de relleno (Padding) en bytes (sin incluir el byte del PaddingSize).
Padding	Byte[*]	Relleno (Padding) añadido al final de cada mensaje para asegurar la longitud de los datos a cifrar sea un entero multiplo del tamaño del bloque de cifrado. El valor de cada byte en el relleno es igual al PaddingSize.
ExtraPaddingSize	Byte	El byte más significativo de un entero de 2 bytes usado para especificar el tamaño del relleno cuando la llave usada para para cifrar el pedazo del mensaje es más grande que 2048 bits. Este campo es omitido si la longitud de la llave es menor o igual a 2048 bits.
Signature	Byte[*]	La firma para el MessageChunk. La firma incluye todas las cabeceras, todos los datos del mensaje, el tamaño del relleno y el relleno.

Esta cola de seguridad contiene la firma del mensaje y se utiliza para ver si los datos recibidos son los mismos que fueron enviados por el remitente, y además, si la identidad de la aplicación que la envía es la del certificado.

En la siguiente figura 2.9 se muestra las diferentes cabeceras de un mensaje en una conversación segura OPC UA sin seguridad captada por medio del programa Wireshark

2015. En esta figura no se aprecia la cola de seguridad del mensaje debido a que la conversación se realizó con la opción de ninguna seguridad.

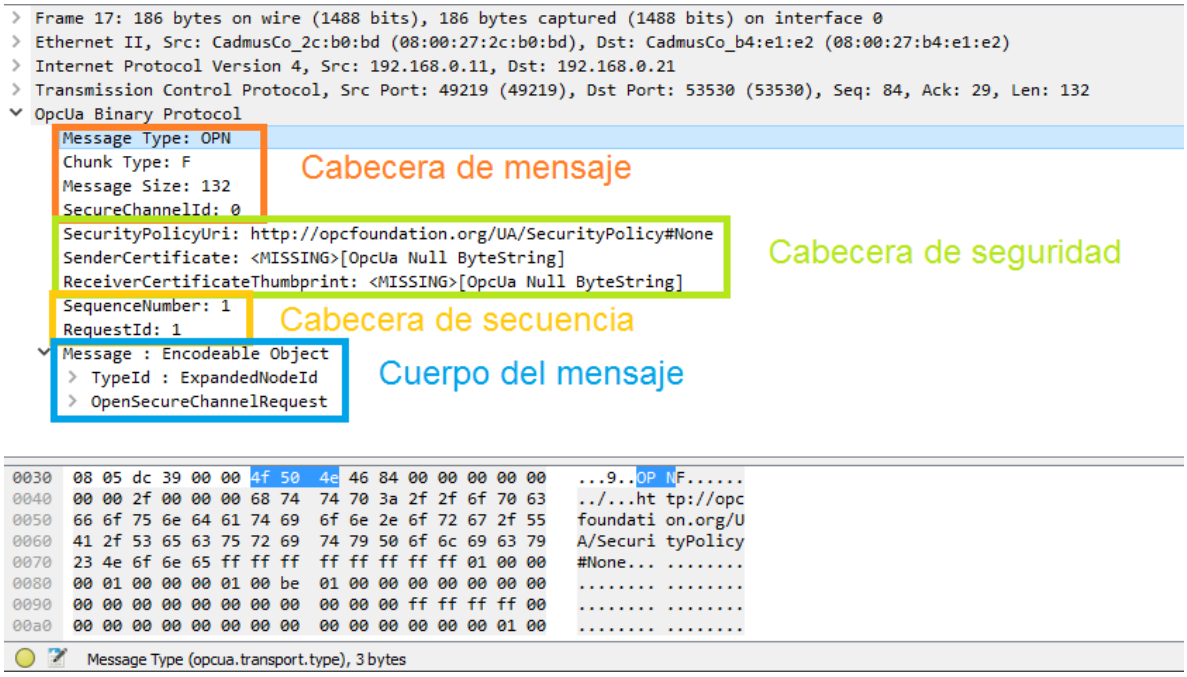

Figura 2.9: **Cabeceras en un mensaje de una UASC obtenida con el programa Wireshark**.

Fuente: Autor.

El cálculo del relleno (Padding), que se muestra en la ecuación 4, depende de la cantidad de datos a enviar a la cual se llama BytesToWrite. El transmisor debe primero calcular la máxima cantidad de bytes disponibles en el pedazo del mensaje, a este cálculo se le denomina MaxBodySize y se obtiene con la siguiente ecuación 2:

$$\text{MaxBodySize} = \text{PlainTextBlockSize} * \text{Floor}\left(\frac{\text{MessageChunkSize} - \text{HeaderSize} - \text{SignatureSize} - 1}{\text{CipherTextBlockSize}}\right) - \text{SequenceHeaderSize}\,.$$

(2-2)

El parámetro HeaderSize incluye el tamaño de la cabecera de Mensaje y la cabecera de Seguridad. El parámetro SequenceHeaderSize es siempre 8 bytes.

Durante el cifrado, un bloque de un tamaño igual a PlainTextBlockSize es cifrado para obtener un bloque del tamaño de CipherTextBlockSize y depende del tipo de algoritmo de cifrado utilizado.

El mensaje cabe en un pedazo (MessageChunk) si el campo BytesToWrite es menor o igual en tamaño a MaxBodySize, en este caso el padding se debe calcular con la siguiente ecuación 3:

$$\text{PaddingSize} = \text{PlainTextBlockSize} - \big((\text{BytesToWrite} + \text{SignatureSize} + 1) \,\%\, \text{PlainTextBlockSize}\big). \quad \textbf{(2-3)}$$

Si el campo BytesToWrite es mayor a MaxBodySize el transmisor debe enviar el mansaje en partes.

Si el pedazo del mensaje (MessageChunk) no está cifrado, el campo PaddingSize y el Padding no aparecen en el mensaje. Si el pedazo del mensaje no está firmado digitalmente el campo de firma no aparece.

Todos los pedazos del mensaje (MessageChunks) son enviados secuencialmente si pertenecen al mismo mensaje. Si hay un error al crear el pedazo del mensaje, el transmisor envía un pedazo final al receptor para que los pedazos anteriormente enviados sean descartados. Para esto utiliza la bandera IsFinal con el valor de "A" que indica abortar. La tabla 2.10 muestra el cuerpo del mensaje de abortar los pedazos anteriores.

Tabla 2.10: Cuerpo del mensaje de abortar en UASC. Adaptado: OPC Unified Architecture, Specification, Part 6.

Name	Data Type	Description
Error	UInt32	Un código numérico para el error.
Reason	String	Una descripción del error. El string usado para la descripción no debe sobrepasar los 4096 caracteres. Un cliente debe ignorar el mensaje si la descripción es mayor del límite.

Un cliente necesita establecer un canal seguro para enviar mensajes, para esto el cliente envía una petición de abrir un canal seguro (OpenSecureChannel) hacia el servidor y este le envía una respuesta. Algunos de los parámetros para establecer el canal seguro se

envían en la cabecera de seguridad y otros en el cuerpo del mensaje. Los parámetros de establecimiento del canal seguro en el cuerpo del mensaje se muestran en la tabla 2.11.

Tabla 2.11: Parámetros de establecimiento de un canal seguro en UASC. Adaptado: OPC Unified Architecture, Specification, Part 6.

Nombre	Tipo de dato
Peticion	
RequestHeader	RequestHeader
ClientProtocolVersion	UInt32
RequestType	SecurityTokenRequestType
SecurityMode	MessageSecurityMode
ClientNonce	ByteString
RequestedLifetime	Int32
Respuesta	
ResponseHeader	ResponseHeader
ServerProtocolVersion	UInt32
SecurityToken	ChannelSecurityToken
SecureChannelId	UInt32
TokenId	UInt32
CreatedAt	DateTime
RevisedLifetime	Int32
ServerNonce	ByteString

Los campos ClientProtocolVersion y ServerProtocolVersion son utilizados para guardar una compatibilidad con futuras actualizaciones. Si el cliente utiliza una versión del protocolo de transporte debe existir una misma versión soportada en el servidor.

El campo RevisedLifeTime indica al cliente cuando debe renovar el Token de seguridad (SecurityToken) por medio de enviar otra petición de un canal seguro. El servidor rechaza nuevas peticiones de renovar el canal seguro si el certificado utilizado por el cliente con el cual fue primeramente solicitado el canal seguro ha cambiado. El cliente puede rechazar el canal si la respuesta está firmada con otro certificado el cual fue utilizado para encriptar la petición.

Los mensajes del establecimiento de canal seguro no son firmados o cifrados si el modo de seguridad es Ninguno, incluso si el modo de seguridad es solamente firmado (SignOnly).

Si el modo de seguridad es Ninguno, no se firma o cifra el mensaje. Los valores de seguridad (Nonces) son ignorados y enviados como nulos. Así no exista seguridad, los

Tokens deben ser renovados, si no se renueva en el tiempo correspondiente, se considera cerrado el canal.

El contenido de cada pedazo del mensaje debe ser descifrado y verificar su firma y el número de secuencia del pedazo. Si se produce un error en el momento de verificación, el receptor debe cerrar el canal de comunicación. Si el receptor es el servidor, este envía un mensaje de error a nivel de transporte antes de cerrar el canal. Si es cerrado el canal, el cliente intenta reabrir el canal y pedir un nuevo token de seguridad con una petición de abrir un nuevo canal seguro. El Servidor debe soportar la política de seguridad utilizada al abrir el canal seguro si se utiliza mecanismos asimétricos de seguridad. El receptor debe validar el certificado del transmisor y enviar un mensaje de error a nivel de transporte si la validación del certificado no se realiza exitosamente. De la misma forma se aplica con la validación de la huella del certificado si no se le reconoce. Si el servidor utiliza seguridad simétrica y no valida el token de seguridad, se debe reportar un error si el token no es validado o ha expirado. Después de validar la seguridad, el receptor debe validar el Id de petición y el número de secuencia. De igual forma, si no se validad debe enviar un error.

2.2.2.2 *Conversación segura WS.*

La conversación segura Web Services (WS) publicado en OASIS07, es una extensión a la especificación WS-Security que especifica conceptos y tecnologías para el intercambio de información de manera segura por medio de servicios Web. OASIS es una organización sin fines de lucro para el establecimiento de estándares abiertos de comunicación.

La conversación segura WS es usada con WS-SecurityPolicy que define los algoritmos de seguridad y con WS-Trust (WS-Trust es una extensión de la especificación de WS que realiza el manejo, expedición y renovación de tokens de seguridad) para la negociación de la compartición de claves seguras para establecer un canal seguro entre aplicaciones OPC UA. WS-SecurityPolicy es utilizada para definir políticas de seguridad y perfiles (los perfiles de seguridad WS, son un grupo de especificaciones de seguridad que cumplen con los requerimientos necesarios establecidos para un determinado objetivo y alcance de seguridad). Para cifrar y firmar datos se utiliza cifrado XML y firmas XML.

La conversación segura WS es optimizada para asegurar datos en XML por medio de cifrado y firmado XML, lo cual le hace útil en el caso de las operaciones a nivel corporativo.

El establecimiento del canal seguro, como la petición de abrir un canal seguro y la respuesta es mapeada a los mensajes RequestSecurityToken (RST) y RequestSecurityTokenResponse (RSTR) de una conversación segura WS.

Actualmente la utilización de conversación segura Web Services no está siendo usada ampliamente en la industria y para enviar datos a niveles corporativos se utiliza Wrappers que trasladan datos de un protocolo como el UASC a otros protocolos de nivel corporativo como RESTful Web Services.

2.2.3 Protocolos de transporte.

2.2.3.1 *Protocolo UA TCP*.

OPC UA TCP es un protocolo simple basado en TCP que establece un canal full dúplex entre un cliente y un servidor. Se diferencia de HTTP porque permite que las respuestas lleguen en cualquier orden y además, si la comunicación falla, se permite el uso de puntos finales de transporte TCP diferentes. Un punto final (Endpoint) en OPC UA debe tener un URL Endpoint, una política de seguridad, un modo de seguridad y el tipo de autenticación. Al referirse que se puede utilizar diferentes puntos finales en OPC UA se quiere decir que se puede usar una combinación diferente de estos 4 campos. Un ejemplo de un Endpoint en OPC UA es:

URL Endpoint = opc.tcp://ServerOpcUA:48001
Política de seguridad = Basic256 (Algoritmo de seguridad utilizado).
Modo de seguridad = Firmado y cifrado.
Tipo de autenticación = Nombre de usuario y contraseña.

Cada mensaje OPC UA TCP tiene una cabecera con campos definidos en la tabla 2.12. El formato de la cabecera OPC UA TCP es idéntico a los primeros 8 bytes de la cabecera de mensaje de UASC, con esto se permite a la capa OPC UA TCP extraer el mensaje proveniente de la capa UASC aun si no se entiende su contenido. La capa OPC

UA TCP verifica el tipo de mensaje y comprueba que el tamaño del mensaje sea menor que el tamaño negociado para los buffers de recepción antes de pasar el mensaje a la capa de UASC.

Tabla 2.12: Campos de la cabecera de mensaje del protocolo OPC UA TCP. Adaptado: OPC Unified Architecture, Specification, Part 6.

Nombre	Tipo	Descripción
MessageType	Byte[3]	Un código ASCII de 3 bytes que identifica el tipo de mensaje. Los siguientes valores son definidos: HEL un mensaje Hello. ACK un mensaje de reconocimiento (Acknowledge). ERR un mensaje de error. La capa de canal seguro define valores adicionales los cuales la capa OPC UA TCP debe aceptar.
Reserved	Byte[1]	Ignorado. Debe ser seteado con el código ASCII de 'F' si el tipo de mensaje es uno de los valores soportados por el protocolo OPC UA TCP.
MessageSize	UInt32	La longitud del mensaje en bytes. Este valor incluye los 8 bytes de la cabecera del mensaje.

El mensaje de tipo Hello y sus campos optativos se muestran en la tabla 2.13.

Tabla 2.13: Campos del mensaje HELLO del protocolo OPC UA TCP. Adaptado: OPC Unified Architecture, Specification, Part 6.

Nombre	Tipo de dato	Descripción
ProtocolVersion	UInt32	La más reciente versión del protocolo OPC UA TCP soportado por el cliente. El servidor podría rechazar al cliente al enviar el mensaje Bad_ProtocolVersionUnsupported. Si el servidor acepta la conexión, este es responsable de asegurar que los mensajes soporten la versión del protocolo. El servidor debe siempre aceptar versiones mayores a la soportada.
ReceiveBufferSize	UInt32	El tamaño del pedazo (MessageChunk) que el transmisor puede recibir. Este valor debe ser más grande que 8 192 bytes.
SendBufferSize	UInt32	El tamaño del pedazo (MessageChunk) que el transmisor puede transmitir. Este valor debe ser más grande que 8 192 bytes.
MaxMessageSize	UInt32	El tamaño máximo para cualquier mensaje de respuesta. El servidor debe abortar el mensaje con un código de estado Bad_ResponseTooLarge, si la respuesta excede el valor. El tamaño del mensaje es calculado usando el cuerpo del mensaje sin encriptar Un valor de cero indica que el cliente no tiene límite.
MaxChunkCount	UInt32	El máximo número de pedazos con cualquier mensaje de respuesta. El servidor debe abortar el mensaje con el código de estado Bad_ResponseTooLarge, si el mensaje de respuesta excede este valor. Un valor de cero indica que el cliente no tiene límite.
EndpointUrl	String	El URL del punto de destino que el cliente desea conectarse. El valor codificado debe ser menor que 4 096 bytes. El servidor debe enviar de regreso un error TcpEndpointUrlInvalid y cerrar la conexión si la longitud excede 4096 o si no se reconoce el recurso identificado por el URL.

El mensaje de reconocimiento (Acknowledge) tiene los campos mostrados en la tabla 2.14.

Tabla 2.14: Campos del mensaje ACKNOWLEDGE del protocolo OPC UA TCP. Adaptado: OPC Unified Architecture, Specification, Part 6.

Nombre	Tipo de dato	Descripción
ProtocolVersion	UInt32	La última versión del protocolo OPC UA TCP soportado por el servidor. Si el cliente acepta la conexión, este es responsable de asegurar que los mensajes soporten la versión del protocolo. El cliente debe siempre aceptar versiones mayores a la soportada.
ReceiveBufferSize	UInt32	El tamaño del pedazo (MessageChunk) que el transmisor puede recibir. Este valor no debe ser más grande que el requerido por el cliente usa en el mensaje Hello. Este valor debe ser más grande que 8 192 bytes.
SendBufferSize	UInt32	El tamaño del pedazo (MessageChunk) que el transmisor puede transmitir. Este valor no debe ser más grande que el requerido por el cliente usa en el mensaje Hello. Este valor debe ser más grande que 8 192 bytes.
MaxMessageSize	UInt32	El tamaño máximo para cualquier mensaje de petición. El servidor debe abortar el mensaje con un código de estado Bad_RequestTooLarge, si la petición excede el valor. El tamaño del mensaje usando el cuerpo de mensaje no encriptado. Un valor de cero indica que el servidor no tiene limite
MaxChunkCount	UInt32	El máximo número de pedazos con cualquier mensaje de petición. El cliente debe abortar el mensaje con el código de estado Bad_RequestTooLarge, si el mensaje de respuesta excede este valor. Un valor de cero indica que el cliente no tiene límite.

El mensaje de error tiene los campos mostrados en la tabla 2.15

Tabla 2.15: Campos del mensaje ERROR del protocolo OPC UA TCP. Adaptado: OPC Unified Architecture, Specification, Part 6.

Nombre	**Tipo**	**Descripción**
Error	UInt32	El código numérico para el error. Cada valor corresponde a un error.
Reason	String	Una mayor descripción del error. El string no debe ser mayor a 4096 caracteres. El cliente debe ignorar strings más grandes que el valor de 4096 caracteres.

En la figura 2.10 se puede observar los campos del mensaje HELLO capturado por el programa Wireshark.

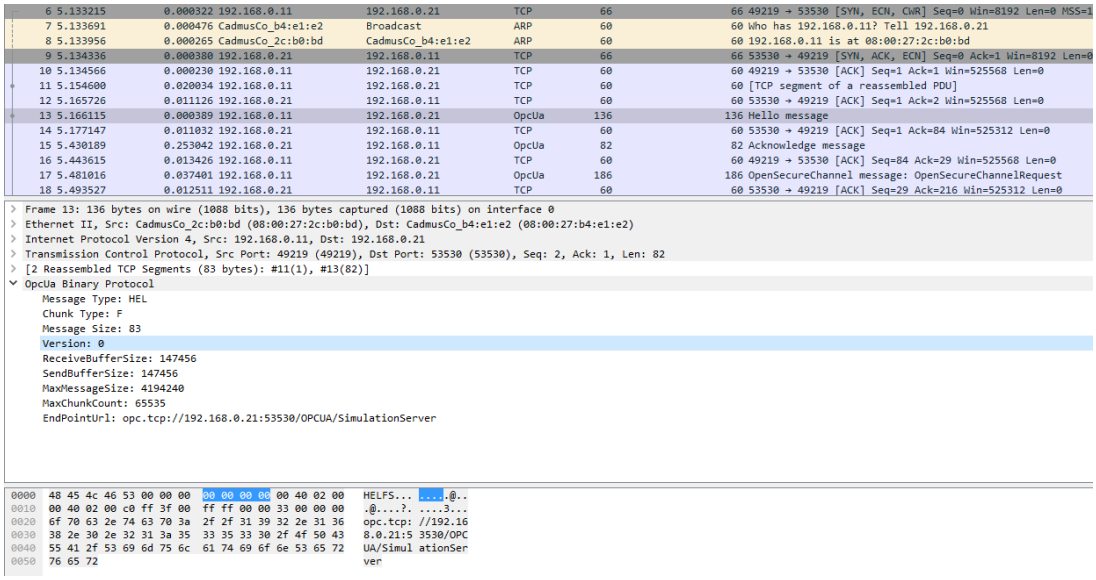

Figura 2.10: Campos de un mensaje HELLO en OPC UA TCP obtenida con el programa Wireshark. Fuente: Autor.

En la figura 2.11 se puede observar los campos del mensaje ACKNOWLEDGE capturado por el programa Wireshark.

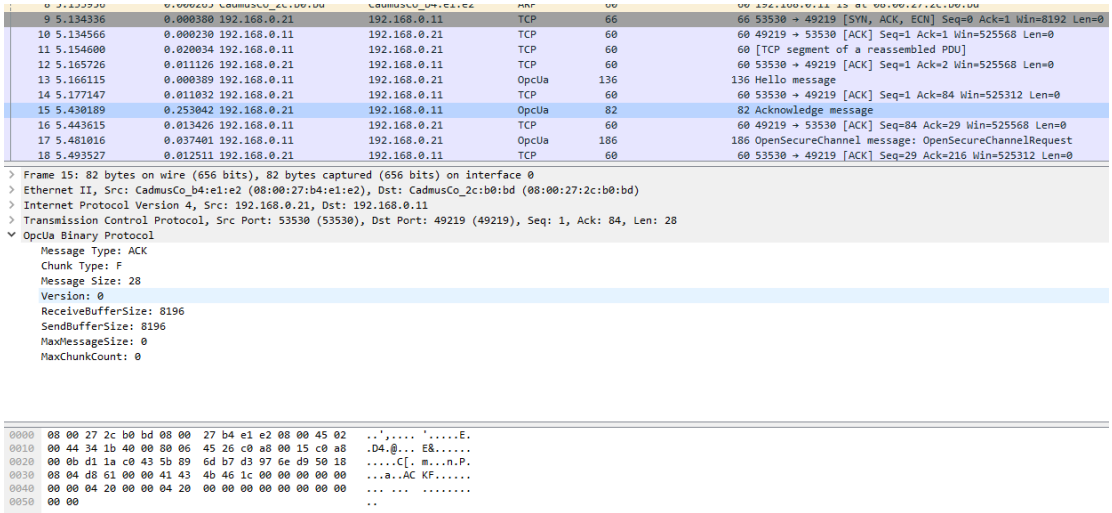

Figura 2.11: Campos de un mensaje ACKNOWLEDGE en OPC UA TCP obtenida con el programa Wireshark. Fuente: Autor.

En la siguiente figura 2.12 se puede observar la estructura de un mensaje, este incluye las cabeceras de cada pedazo (MessageChunk), la firma y el relleno de cola

(footer). Además, se muestra que las cabeceras y el relleno forman parte de los datos firmados, y que la cabecera de mensaje y seguridad no es encriptada.

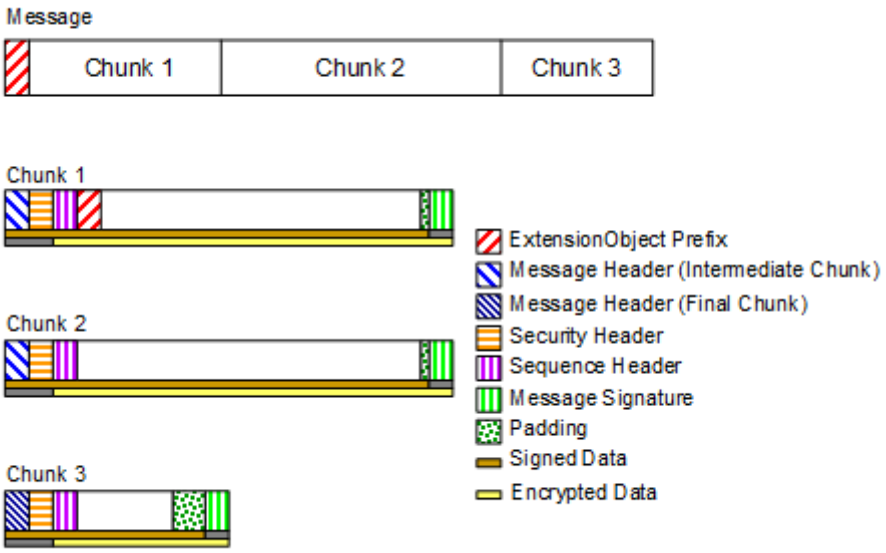

Figura 2.12: Estructura de un mensaje OPC UA TCP. Fuente: OPC Unified Architecture, Specification, Part 6.

El establecimiento de una conexión OPC UA es siempre iniciada por un cliente el cual crea un socket para luego enviar una petición de abrir un canal seguro. Después de crear el socket envía un mensaje HELLO con los buffers que el cliente soporta. El servidor responde con un mensaje ACKNOWLEDGE y se completa la negociación de los tamaños de los buffers. El tamaño de los buffers es reportado a la capa de canal seguro. Además, se especifica el tamaño de los pedazos (MessageCHunks) a enviar en la conexión.

El envío de los mensajes Hello y Acknowledge es realizado una sola vez, si se ejecutan otra vez, se produce un error y se cierra el socket. Los servidores cierran el socket si no reciben un mensaje Hello en un determinado tiempo que por defecto es 2 minutos.

El cliente envía una petición de canal seguro y recibe la respuesta del servidor una sola vez. El servidor asocia al socket con el Id del canal seguro, y el cliente hace lo mismo al recibir la respuesta.

La secuencia de mensajes cuando se establece una comunicación OPC UA TCP se muestra en la figura 2.13

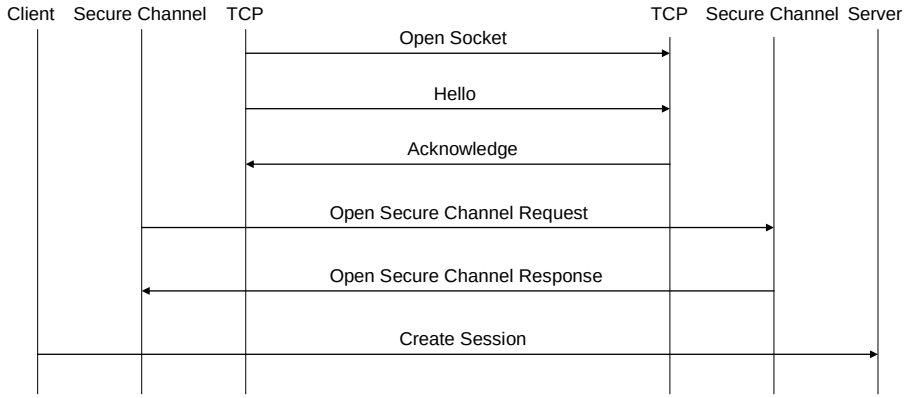

Figura 2.13: Establecimiento de una conexión OPC UA TCP. Fuente: OPC Unified Architecture, Specification, Part 6.

El cliente es el que cierra la conexión por medio de una petición CloseSecureChannel y cerrando el socket asociado. El servidor no envía una respuesta de la petición de cierre de canal seguro, solo libera los recursos asociados al canal. Si la verificación de seguridad de la petición de cierre de canal seguro falla, el servidor cierra el socket y permite la reconexión del cliente. La figura 2.14 muestra la secuencia de cierre de canal seguro.

Cuando se produce un error, el servidor debe enviar un mensaje de error al cliente y cerrar el socket. Si el cliente encuentra un error únicamente cierra el socket y no envía al servidor un mensaje de error. Después de que se cierra el socket, el cliente trata de reconectarse automáticamente.

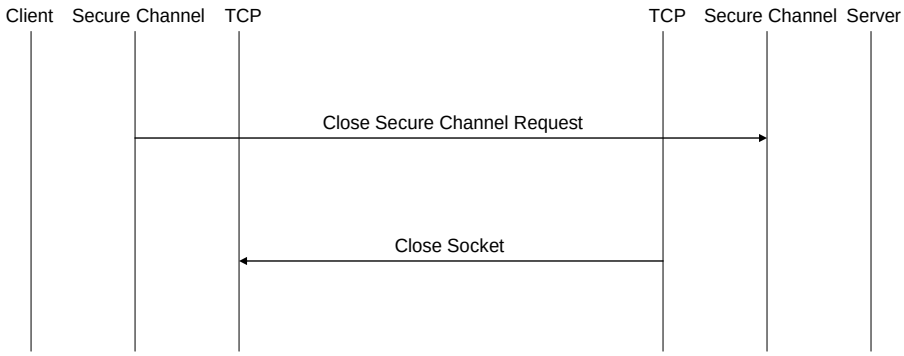

Figura 2.14: Cierre de una conexión OPC UA TCP. Fuente: OPC Unified Architecture, Specification, Part 6.

La tabla 2.16 muestra los posibles códigos de error en la comunicación OPC UA TCP.

Cuando el cliente establece un canal seguro, este entra en modo de recuperación de errores al cerrarse inesperadamente el socket o si el servidor envía un mensaje de error OPC UA TCP. Mientras el cliente está en este estado, trata de reconectarse al encontrar error. Si la reconexión es exitosa, el cliente envía un mensaje Hello y envía una petición de canal seguro con la re autenticación del cliente y por último se asocia el nuevo socket con el canal seguro reconectado.

Cuando se realiza la primera reconexión, esta debe ser inmediata y para la siguiente debe esperar un segundo. Este tiempo aumentara con cada intento de reconexión hasta un máximo de 2 minutos. El periodo de espera para cada reconexión debe ser el doble del anterior periodo de espera. Se intenta la reconexión hasta cerrar el canal seguro después de un periodo establecido en el campo RevisedLifeTime cuando expira el último SecurityToken. El servidor con su stack debe tener un mecanismo de responder y esperar la petición del cliente cuando se crea el nuevo socket. El cliente no debe anular peticiones que han sido enviadas y que esperan respuesta cuando el socket es cerrado. Sin embargo existe un tiempo de espera que reporta este caso a la aplicación.

Tabla 2.16: Campos del mensaje ERROR del protocolo OPC UA TCP. Adaptado: OPC Unified Architecture, Specification, Part 6.

Nombre	Descripción
TcpServerTooBusy	El servidor no procesa la petición porque está ocupado. Pertenece al servidor cuando este necesita retornar el mensaje. Un servidor puede controlar como frecuentemente un cliente se reconecta al esperar que se regrese este error.
TcpMessageTypeInvalid	El tipo de mensaje especificado en la cabecera es invalido. Cada mensaje comienza con una secuencia de valores ASCII de 4 bytes que identifican el tipo de mensaje. El servidor regresa este error si el tipo de mensaje no es aceptado. Algunos de los tipos de mensajes son definidos en la capa de canal seguro.
TcpSecureChannelUnknown	El SecureChannelId o el TokenId no están actualmente en uso. Este error es reportado por la capa de canal seguro.
TcpMessageTooLarge	El tamaño del mensaje especificado en la cabecera es demasiado largo. El servidor regresa este erro si el tamaño del mensaje excede el tamaño máximo del buffer de recepción negociado .
TcpTimeout	Tiempo de espera agotado al acceder a un recurso. El servidor determina cuando expira el tiempo de espera.
TcpNotEnoughResources	No hay suficientes recursos para atender la petición. El servidor regresa este error si está corriendo con falta e memoria o de algún recurso similar. Un servidor puede controlar cuando un cliente se reconecta al esperar este error.
TcpInternalError	Un error interno ha ocurrido Este debería aparecer si un error de programación o de configuración ocurre.
TcpEndpointUrlInvalid	El servidor no reconoce el Uri especificado.
SecurityChecksFailed	El mensaje fue rechazado porque no pudo ser verificado.
RequestInterrupted	La petición no pudo ser enviada por una interrupción en la red.
RequestTimeout	Tiempo de espera agotado mientras se procesaba la petición.
SecureChannelClosed	El canal seguro se ha cerrado.
SecureChannelTokenUnknown	El token de seguridad ha expirado o no es reconocido.
CertificateUntrusted	El certificado del transmisor no es confiable para el receptor.
CertificateTimeInvalid	El certificado del transmisor ha expirado o ya no es válido.
CertificateIssuerTimeInvalid	El publicante del certificado del transmisor ha expirado o no es válido.
CertificateUseNotAllowed	El certificado del transmisor no debe ser usado para establecer el canal seguro.
CertificateIssuerUseNotAllowed	El certificado del publicante no debe ser usado como autoridad del certificado.
CertificateRevocationUnknown	No puede ser verificado el estado de revocación del certificado del transmisor.
CertificateIssuerRevocationUnknown	No puede ser verificado el estado de revocación del certificado del publicante.
CertificateRevoked	El certificado del transmisor ha sido revocado por el publicante.
IssuerCertificateRevoked	El certificado del publicante ha sido revocado por el publicante.
CertificateUnknown	La huella digital del certificado del receptor no es reconocida por el receptor.

Si el cliente envía una nueva petición, esta debe ser buferizada o retornar un mensaje de error de Bad_TcpRequestInterrupted. El cliente detiene el proceso de reconexión cerrando el canal seguro.

Si servidor debe abandonar el canal seguro antes de que el cliente sea capaz de reconectarse, este debe enviar un mensaje de error Bad_TcpSecureChannelUnknow cuando el cliente envía una petición de abrir canal seguro, este error es enviado a la aplicación para crear un nuevo canal seguro.

Los buffers negociados se mantienen a menos de que se cree un nuevo socket para un nuevo canal seguro, si esto pasa se debe enviar un mensaje de error Bad_TcpSecureChannelClose a la aplicación.

La figura 2.15 muestra la secuencia de mensajes asociados a la recuperación ante errores en la conexión OPC UA TCP.

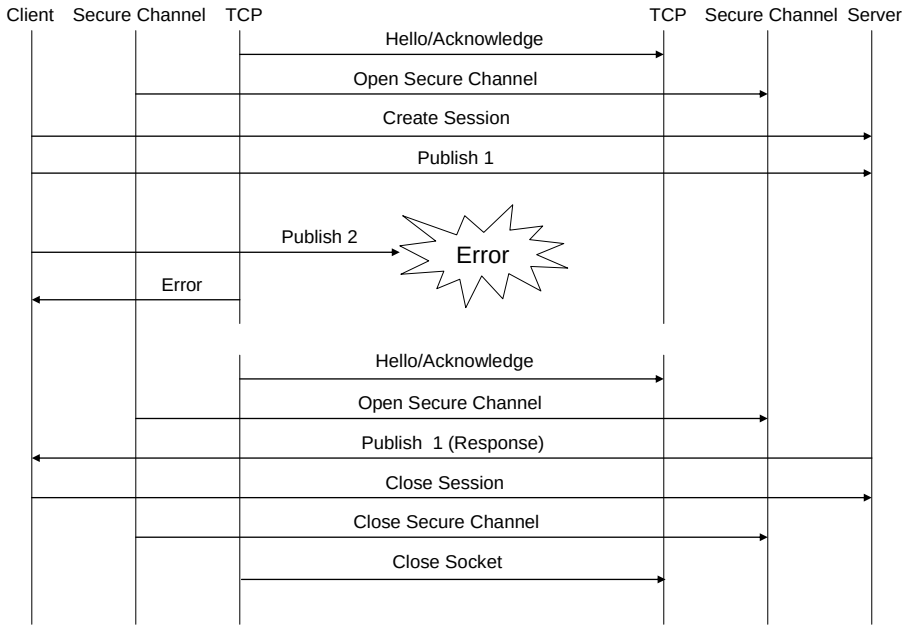

Figura 2.15: Recuperación ante errores en OPC UA TCP. Fuente: OPC Unified Architecture, Specification, Part 6.

2.2.3.2 *Protocolo HTTPS.*

HTTPS son mensajes HTTP enviados sobre una conexión SSL/TLS. Los mensajes no cambian únicamente en lugar de usar una conexión TCP/IP se usa una conexión TLS.

HTTPS es un protocolo que provee de seguridad, es decir todos los bytes de los mensajes son seguros. HTTPS no permite intermediarios o proxies para manejar tráfico, es decir es una comunicación punto a punto.

Una cabecera HTTP llamada "OPCUA-SecurityPolicy" es usada por el cliente para avisar al servidor que política de seguridad que se usara de las posibles opciones. El valor de la cabecera es el URI de la política de seguridad. Si se omite la cabecera, se entiende que la política de seguridad es ninguna.

Todas las comunicaciones HTTPS a través del URL son tratadas como una única conexión de canal seguro que es compartida por muchos clientes.

Los algoritmos de cifrado usados por HTTPS no se relacionan a las políticas de seguridad de OPC UA.

En algunos casos HTTPS confía en intermediarios que algunas aplicaciones consideran inseguras, en este caso se debe optar por una conexión VPN.

Las aplicaciones que soportan el transporte HTTPS deben soportar HTTP 1.1 y SSL/TLS 1.0.

En algunas implementaciones HTTPS se necesita de un certificado que tenga el nombre DNS asociado a la máquina que aloja el servidor. A veces se tiene varios nombres DNS de varios servidores alojados en la máquina y comparten varios certificados HTTPS. Por lo tanto no es lo mismo el certificado HTTPS y el certificado de aplicación. Por lo tanto la aplicación que use transporte HTTPS y requiere de autenticación de aplicación debe revisar el certificado de la aplicación.

En la figura 2.16 se muestra los posibles escenarios del transporte con HTTPS.

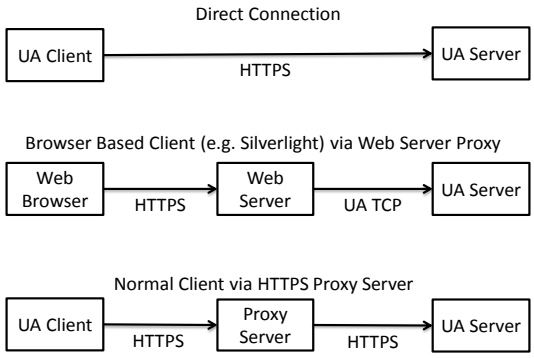

Figura 2.16: Escenarios de uso del transporte HTTPS en OPC UA. Fuente: OPC Unified Architecture, Specification, Part 6.

Al utilizar mensajes HTTPS se puede utilizar codificación XML o codificación binaria OPC UA. Al usar codificación XML se implementa los servicios OPC con un formato mensaje de petición-respuesta SOAP (Simple Object Access Protocol) sobre la conexión HTTPS. Ver figura 2.6.

El cuerpo del mensaje HTTP puede ser un mensaje SOAP 1.2. La codificación OPC UA XML puede representar un mensaje OPC UA como un elemento XML. Este elemento es añadido al mensaje SOAP como el único hijo del cuerpo SOAP. Si un error ocurre en el servidor mientras se analiza el cuerpo de la petición, el servidor debe retornar una falla SOAP o retornar una respuesta como error OPC UA. En la siguiente figura 2.17 se muestra el encapsulamiento de la petición OPC UA en un mensaje SOAP.

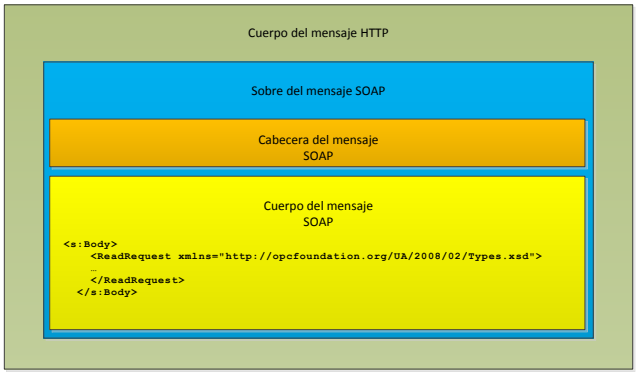

Figura 2.17: Encapsulamiento del mensaje SOAP en el cuerpo del mensaje HTTP. Fuente: autor.

La acción SOAP asociada con la petición codificada como XML siempre tendrá la forma: http://opcfoundation.org/UA/2008/02/Services.wsdl/<service name> donde service name es el nombre del servicio OPC UA utilizado.

La acción SOAP asociada con la respuesta codificada como XML siempre tendrá la forma: http://opcfoundation.org/UA/2008/02/Services.wsdl/<service name>Response donde "service name" es el nombre del servicio OPC UA utilizado.

Todas las peticiones deberán ser peticiones HTTP POST. El tipo-contenido deberá ser "application/soap+xml", el set de caracteres y parámetros de acción deben ser especificados. Los parámetros del set de caracteres debe ser "utf-8" y el parámetro de acción deberá ser el URI de la acción SOAP.

En las líneas a continuación se observa la cabecera de una petición HTTP y de un ejemplo de un mensaje de petición.

```
POST /UA/SampleServer HTTP/1.1
Content-Type: application/soap+xml; charset="utf-8";
    action="http://opcfoundation.org/UA/2008/02/Services.wsdl/Read"
Content-Length: nnnn

<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Body>
    <ReadRequest xmlns="http://opcfoundation.org/UA/2008/02/Types.xsd">
    …
    </ReadRequest>
  </s:Body>
</s:Envelope>
```

En las siguientes líneas se observa la cabecera de una respuesta HTTP y de un ejemplo de un mensaje de respuesta.

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8";
    action="http://opcfoundation.org/UA/2008/02/Services.wsdl/ReadResponse"
Content-Length: nnnn

<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Body>
    <ReadResponse xmlns="http://opcfoundation.org/UA/2008/02/Types.xsd">
    …
    </ReadResponse>
  </s:Body>
</s:Envelope>
```

A más de codificación XML se tiene codificación binaria OPC UA de los mensajes que pueden ir sobre una conexión HTTPS. Si la aplicación soporta el transporte

HTTPS debe soportar HTTP 1.1. El cuerpo del mensaje HTTP debe ser un blob (binary large object) codificado en forma binaria OPC UA. El contenido-tipo del mensaje debe ser “application/octet-stream”.

En las siguientes líneas se puede observar una cabecera de petición y de respuesta con codificación binaria OPC UA con HTTPS.

```
POST /UA/SampleServer HTTP/1.1
Content-Type: application/octet-stream;
Content-Length: nnnn


HTTP/1.1 200 OK
Content-Type: application/octet-stream;
Content-Length: nnnn
```

2.3 SEGURIDAD.

2.3.1 Modelo de seguridad en OPC UA.

El Protocolo OPC UA puede funcionar en los 3 niveles de operación de una empresa. El primer nivel de planta es una red que comunica controladores industriales con sistemas de supervisión y el intercambio de información debe ser rápido y seguro. El siguiente nivel de operaciones de la empresa es una red que maneja datos de producción y en este nivel la seguridad toma más importancia. El último nivel en donde están los sistemas ERP está los datos e indicadores más importantes de la empresa y la seguridad es primordial, inclusive porque se tiene acceso remoto el cual puede ser de internet hacia este nivel. Dependiendo del nivel de uso del protocolo OPC UA se necesita diferentes modelos de seguridad y desempeño. En la siguiente figura 2.18 se puede observar los 3 niveles de uso del protocolo OPC UA.

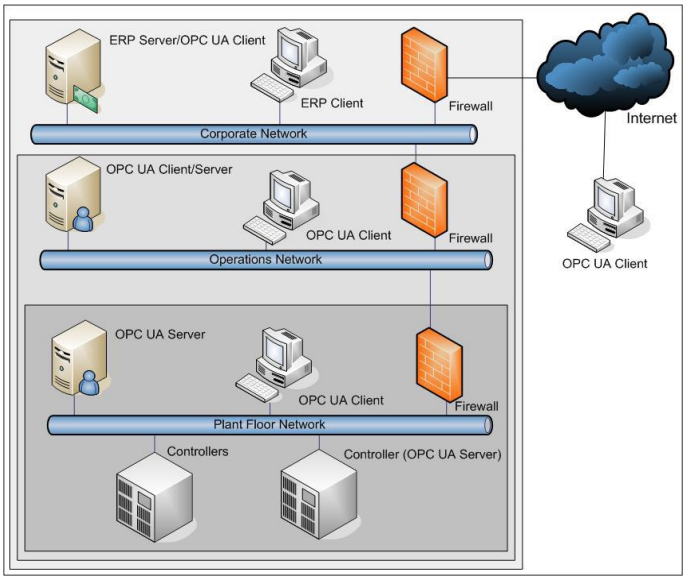

Figura 2.18: Niveles de uso de OPC UA en una empresa. Adaptado: OPC Unified Architecture. Wolfgang Mahnke. 2009.

Cada capa de la aplicación OPC UA tiene responsabilidades en seguridad. La capa de aplicación utiliza una sesión para intercambiar datos y en esta capa se realiza la autenticación y autorización de usuarios así como la de productos. La sesión se establece después de que se haya establecido un canal seguro.

La siguiente capa es la de comunicación en donde se establece un canal seguro. Esta capa es responsable de la autenticación y autorización de aplicaciones, esto se realiza por medio de certificados X.509. Para la creación y la verificación se puede realizar por medio de una infraestructura o por la misma aplicación. En esta capa también se realiza la integridad de la información por medio del firmado digital y la confidencialidad por medio del cifrado.

La capa de transporte es responsable de transmitir y recibir información mediante la conexión por medio de sockets. En esta capa existen mecanismo para la recuperación ante errores y de ataques como DOS (Denial of Services).

En la capa de aplicación se puede realizar la autorización y autenticación a nivel de producto, en donde un producto o una determinada versión de un producto se certifican por medio de certificados especiales otorgados por una Autoridad de

Certificación (CA). Para otorgar los certificados especiales, el CA realiza pruebas en un laboratorio de certificación con referencia a los perfiles que indica que cumple el producto.

En la figura 2.19 se muestra las diferentes capas de OPC UA y su responsabilidad en seguridad.

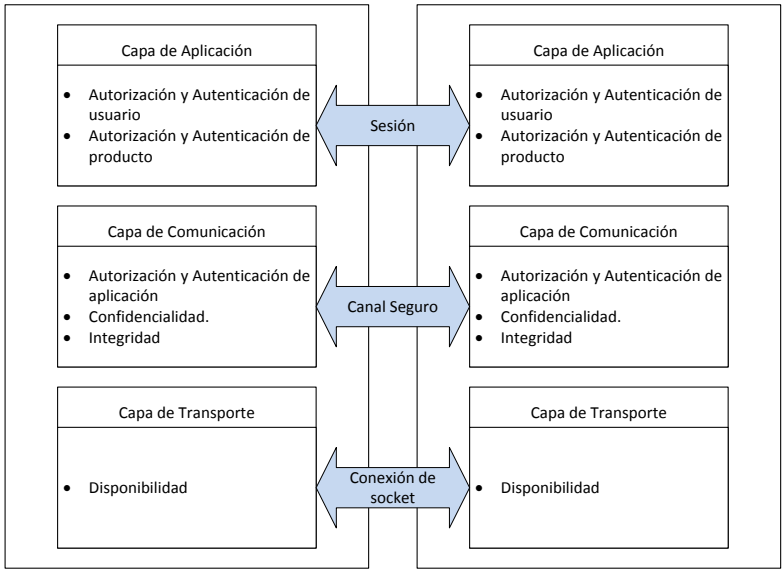

Figura 2.19: Responsabilidades de seguridad en OPC UA. Adaptado: OPC Unified Architecture. Wolfgang Mahnke. 2009.

En la siguiente figura 2.20 se muestra la secuencia de mensajes para establecer un canal seguro. Un cliente si no sabe las características que expone el servidor debe enviar una petición de los puntos finales (Endpoints) que indican la configuración de seguridad es decir las políticas de seguridad, los modos de seguridad, las políticas del token de seguridad y los certificados a intercambiar de la aplicación. Al recibir la respuesta de los puntos finales del servidor, se escoge uno de ellos y se realiza una validación del certificado de la aplicación por medio de una autoridad de validación (Validation Authority VA) la cual emite una respuesta si el certificado es validado. Posteriormente a la validación, el cliente envía la petición del canal seguro al servidor que es firmado con la clave pública del cliente y encriptado con la llave privada del servidor, después el servidor envía una respuesta, la cual debe ser verificar en el cliente con el mismo procedimiento. Después de establecer el canal seguro por medio de algoritmos asimétricos se negocia los futuros mensajes con claves simétricas, esto debido al extenso procesamiento que se tienen en los

mecanismos asimétricos. Al terminar un tiempo establecido, el canal seguro expira y debe iniciarse nuevamente el proceso.

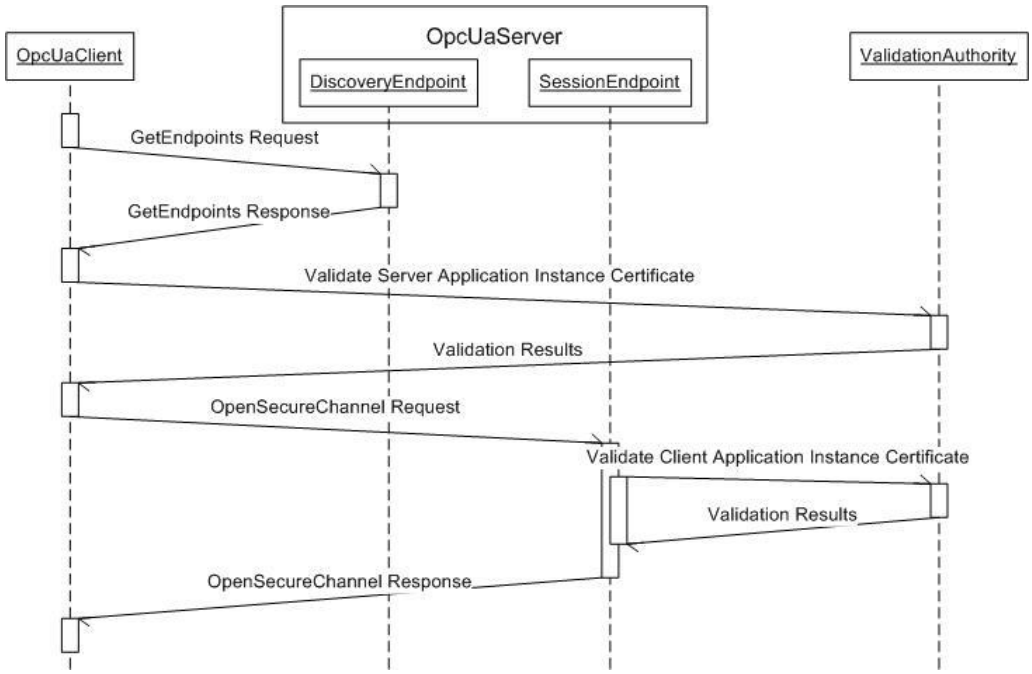

Figura 2.20: Establecimiento de un canal seguro con validación de certificados del cliente en OPC UA. Adaptado: OPC Unified Architecture. Wolfgang Mahnke. 2009.

El siguiente paso es la creación y activación de la sesión. Una vez que el cliente recibe la respuesta del servidor de tener el canal seguro, el cliente valida el certificado del servidor con el Validation Authority (VA) y al recibir la respuesta se valida al servidor que le indica las capacidades y funcionalidades que tiene el servidor.

El último paso es la activación de la sesión en la cual se envía la información de seguridad de los usuarios y sus credenciales con el certificado del cliente. Las credenciales pueden estar en forma de certificado X.509 o como usuario y contraseña. En la figura 2.21 se muestra el proceso de creación y activación de sesión.

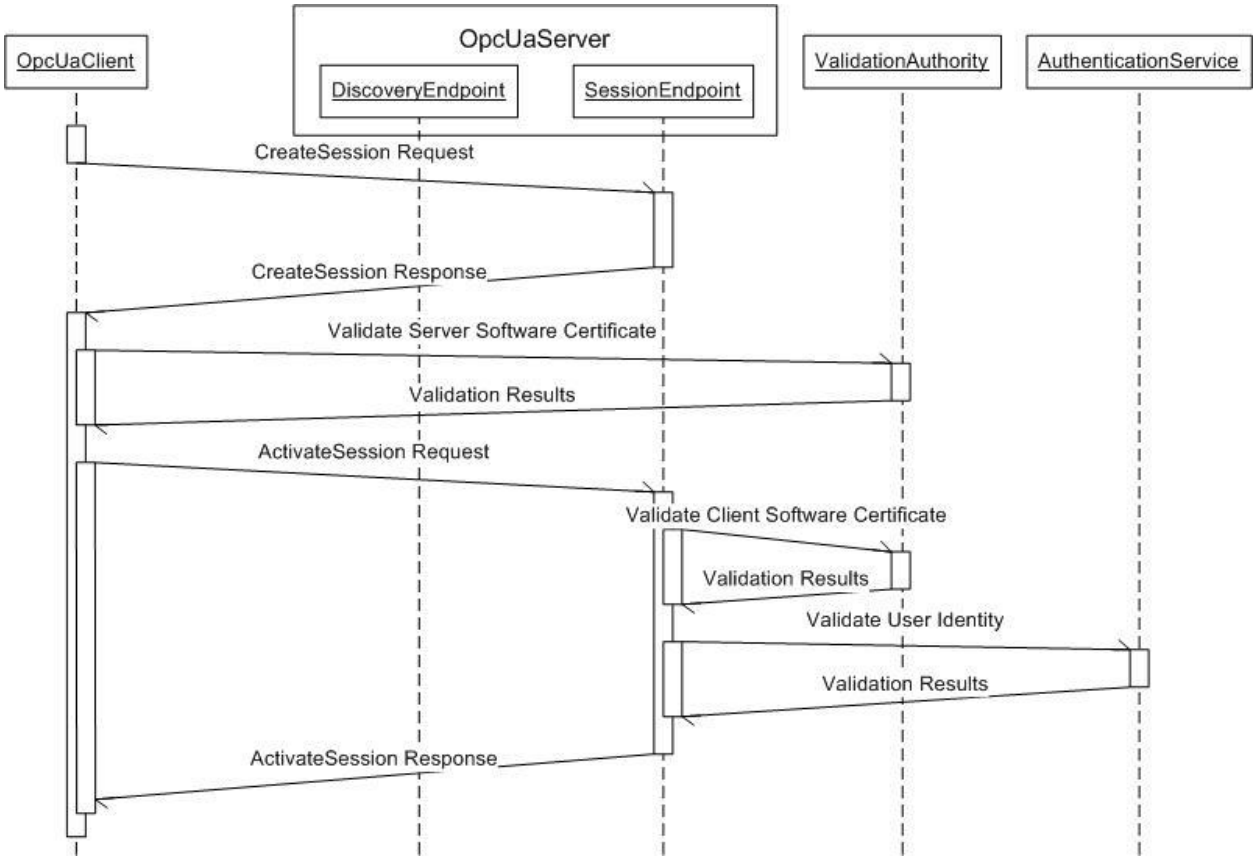

Figura 2.21: Establecimiento de una sesión con validación de usuarios en OPC UA. Adaptado: OPC Unified Architecture. Wolfgang Mahnke. 2009.

En la tabla 2.17 se puede ver las diferentes opciones en OPC UA de los tipos de credenciales para activar la sesión.

Tabla 2.17: Tipos de credenciales de usuario en OPC UA. Adaptado: OPC Unified Architecture. Wolfgang Mahnke. 2009.

ID Simbólico	Descripción
Anonymous	No se envía información del usuario.
UserName	Un usuario identificado por nombre y contraseña
X509v3	Un usuario identificado por un certificado X509v3
WSS	Un usuario identificado por un WS-SecurityToken. (e.g. SAML,Kerberos-Ticket)

2.3.2 Certificados OPC UA.

Un certificado digital es un documento electrónico principalmente usado para distribuir llaves públicas o a un par de privada y pública que son comprobadas por una entidad externa. OPC UA principalmente se basa en certificados X.509V3. En la siguiente tabla 2.18 se observa el contenido de un certificado X509V3.

Los certificados pueden ser autofirmados (Self-signed) y firmados por una autoridad certificadora confiable (CA). El ciclo de vida de los certificados es administrado por una infraestructura de llave pública. La figura 2.22 muestra las diferencias entre un certificado autofirmado y certificado por un CA.

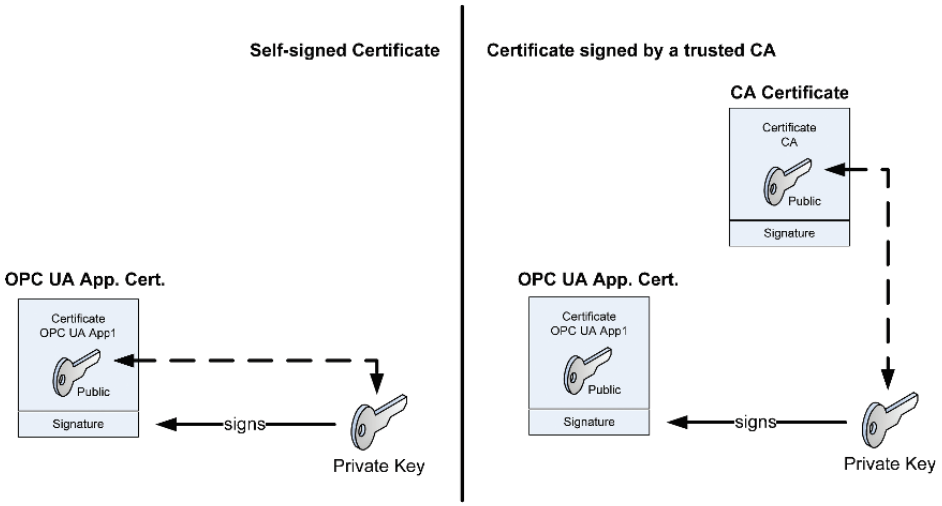

Figura 2.22: Certificados autofirmados y firmados por un CA. Fuente: OPC Unified Architecture. Wolfgang Mahnke. 2009.

Tabla 2.18: Contenido de un certificado X.509V3. Adaptado: OPC Unified Architecture. Wolfgang Mahnke. 2009.

Version	Describe la versión del certificado que puede ser 1,2, o 3. Para OPC UA esta debe ser 3
Serial Number	Debe ser un numero positivo que identifica al certificado otorgado por un CA particular y debe ser único.
Signature Algorithm	Contiene el identificador del algoritmo de firma usado por el CA para firmar el certificado.
Issuer	Identifica al CA que otorgo y firmo el certificado. El identificador del CA es representado por un distinguished name (DN)
Valid From	La fecha cuando el periodo de validación del certificado comienza.
Valid To	La fecha cuando el periodo de validación del certificado termina.
Subject	El identificador de la entidad que pertenece el certificado presentado con un distinguished name (DN). Una extensión V3 puede ser dada como un nombre alternativo (subjectAlternativeName) para esta entidad la cual debe proveer información adicional o simplemente que sea más entendible para humanos.
Public Key	Contiene el identificador del tipo de llave publica del dueño del certificado así como la llave mismo.
<Extensions>	Extensiones son solo disponibles cuando se usa la versión 3 de los certificados X.509V3. Extensiones estándares son por ejemplo uso de la llave, políticas del certificado, nombres alternativos y puntos de distribución de CRL (Lista de revocación de certificados).
Signature	Contiene la firma digital creada por el CA para firmar el certificado.

En OPC UA se utilizan 3 tipos de certificados: Certificado de instancia de la aplicación, certificado de software y certificado de usuario.

Los certificados OPC UA de instancia de aplicación se requiere con cada instalación de una aplicación. Este certificado identifica a una instancia corriendo la aplicación instalada, el cual puede ser autofirmado o firmado por un CA. La tabla 2.19 muestra los contenidos de un certificado OPC UA de instancia de aplicación.

Otro de los certificados usados en OPC UA son los de software, estos se diferencian de los certificados de instancia de aplicación que son usados para determinar una versión de un producto OPC UA. Este tiene una extensión V3 que indica los perfiles OPC UA probados y que pasaron la prueba. Este certificado se obtiene al poner a prueba un producto en un laboratorio acreditado en certificación OPC UA.

Tabla 2.19: Extensiones V3 de un certificado de instancia de aplicación OPC UA. Adaptado: OPC Unified Architecture. Wolfgang Mahnke. 2009.

SubjectAltName (Ext)	Los nombres alternativos para la instancia de la aplicación deben incluir un URI (uniformResourceIdentifier) el cual es igual al Uri de la aplicación. Los servidores deben especificar un nombre DNS o una dirección IP la cual identifica la maquina donde la instancia de la aplicación corre. Nombres DNS adicionales pueden ser especificados si la maquina tiene múltiples nombres. La dirección IP no debe ser especificada si el servidor tiene nombre DNS.
Key Usage (Ext)	Especifica como la llave del certificado debe ser usada. Debe incluir los siguientes casos: digitalSignature, nonRepudiation, keyEncipherment y dataEncipherment. Otros usos pueden ser especificados.
Extended Key Usage (Ext)	Especifica usos adicionales de la llave del certificado. Debe especificar serverAuth y/o clientAuth. Otros usos pueden ser especificados.

Tabla 2.20: Extensiones V3 de un certificado de software OPC UA. Adaptado: OPC Unified Architecture. Wolfgang Mahnke. 2009.

SubjectAltName (Ext)	Los nombres alternativos para el producto debe incluir un URI (uniformResourceIdentifier) el cual es igual al Uri del producto.
Key Usage (Ext)	Especifica como la llave del certificado debe ser usada. Debe incluir los siguientes casos: digitalSignature, nonRepudiation, keyEncipherment y dataEncipherment. Otros usos pueden ser especificados.
Extended Key Usage (Ext)	Especifica usos adicionales de la llave del certificado. Debe especificar "codeSigning". Otros usos pueden ser especificados.
softwareCertificate	La forma codificada en XML del certificado del software guardado en text UTF8.

2.3.3 Infraestructura pública de llave para OPC UA.

Una infraestructura de llave pública (PKI) es utilizado para el manejo de certificados digitales y posee diferentes entidades con roles específicos como la autoridad de registro (RA), la autoridad de certificación (CA), la autoridad de validación (VA) y las entidades finales (EE). El CA otorga, renueva y revoca a los certificados y reporta a otras entidades de las diferentes acciones realizadas.

En un PKI existen dos modelos de confianza: el modelo directo y el modelo jerárquico.

En el modelo directo, un cliente OPC UA confía en un servidor OPC UA si confía en que la llave pública del certificado del servidor es en realidad la del servidor. Esto se lleva a cabo si el administrador decide que el certificado del servidor OPC UA deber ser confiable y guardado en la base de datos del cliente que contiene a todos los certificados confiables. La figura 2.23 muestra el modelo directo de confianza en OPC UA.

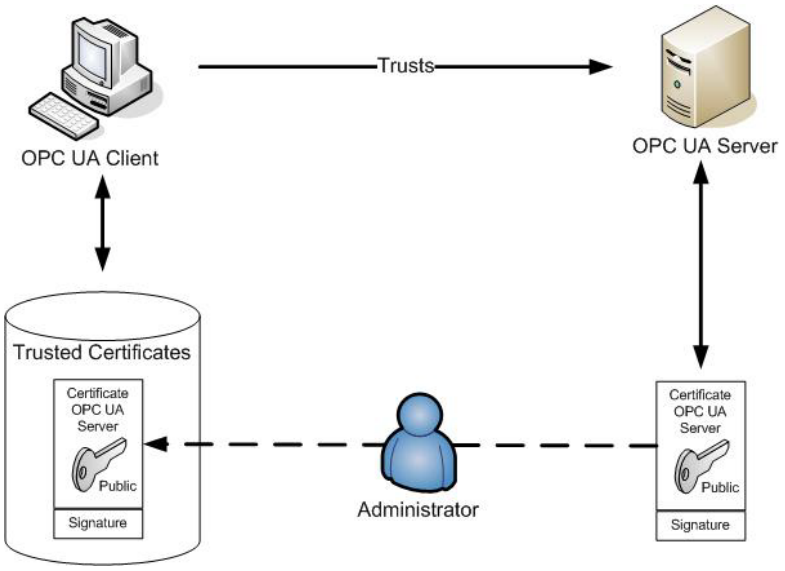

Figura 2.23: Modelo directo de confianza en OPC UA. Fuente: OPC Unified Architecture. Wolfgang Mahnke. 2009.

En el modelo jerárquico el certificado del servidor es firmado por un CA confiable. En lugar de guardar el certificado del servidor, se guarda el certificado del CA en la base de datos del cliente. El cliente confía en el certificado del servidor porque confía en el certificado del CA. En la figura 2.24 se muestra el modelo de confianza jerárquico.

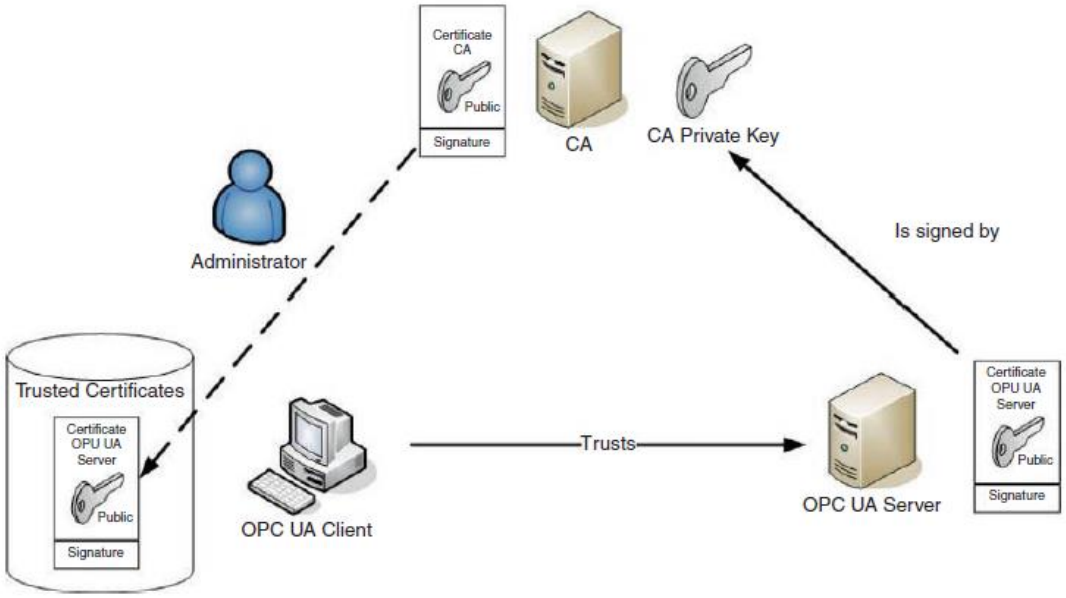

Figura 2.24: Modelo de confianza jerárquico en OPC UA. Fuente: OPC Unified Architecture. Wolfgang Mahnke. 2009.

2.4 ARQUITECTURA DE APLICACIÓN OPC UA.

Al momento de realizar una aplicación para que utilice OPC UA debe tratar con dos capas, la capa del SDK (Software Developmente Kit) en donde esta implementado las funciones de alto nivel y el stack en donde se encuentran las funciones de bajo nivel. Existen muchos fabricantes que han desarrollado SDKs para diferentes plataformas, como por ejemplo el desarrollador de productos PROSYS, MATRIKON, SOFTING entre algunos. El SDK es proveído por la fundación OPC a sus miembros.

2.4.1 stack o Pila de OPC UA.

El stack es aquel que cubre las funciones de bajo nivel, entre estas están la capa de software de codificación, la capa de software que implementa la seguridad, la capa de transporte, y la capa de plataforma. En medio del SDK y el stack están las APIs de cliente y servidor. En la siguiente figura 2.25 se muestra las capas del SDK.

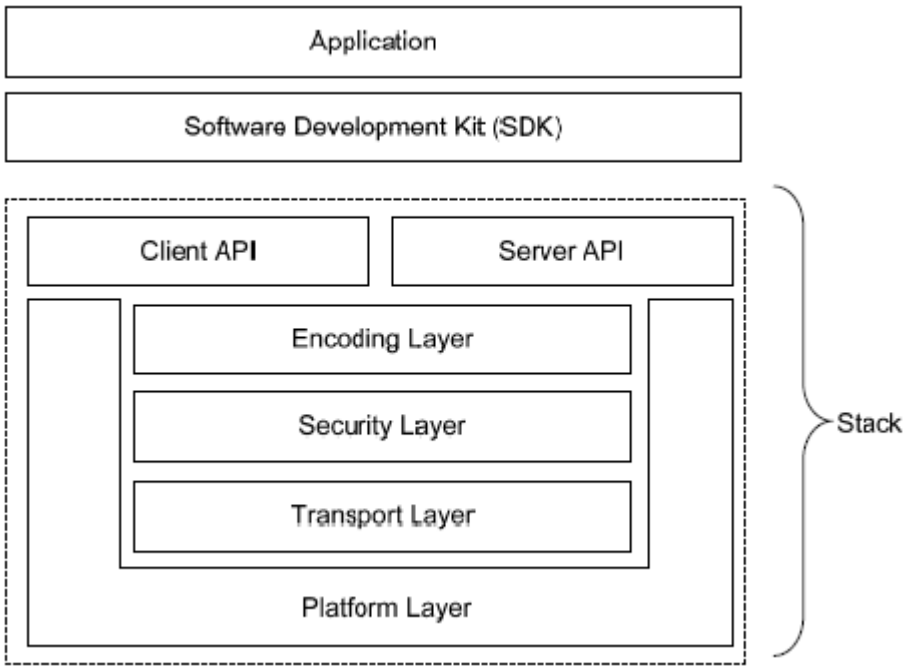

Figura 2.25: Capas de software del stack OPC UA. Fuente: OPC Unified Architecture. Wolfgang Mahnke. 2009.

2.4.1.1 *Interfaces.*

Tanto el cliente como el servidor tienen APIs diferentes ya que cada uno tiene funciones diferentes. El servidor envía los datos como respuesta a las peticiones y subscripciones de un cliente.

2.4.1.2 *Capa de Codificación.*

En esta capa se debe codificar y decodificar los mensajes. Los mensajes de servicios son trasladados desde la capa de las APIs y son serializados en esta capa para luego enviarlos a la capa de seguridad.

2.4.1.3 *Capa de seguridad.*

Los mensajes de los servicios ya codificados y serializados son asegurados en esta capa. En esta capa se ofrecen funciones de cifrado y firmado digital. En esta capa se analizan las cabeceras y colas de seguridad.

2.4.1.4 *Capa de transporte.*

Esta capa es responsable de transmitir y recibir mensajes y tratar errores de red. En esta capa se analizan cabeceras para conocer el tipo y longitud del mensaje recibido o transmitido.

2.4.1.5 *Capa de Plataforma.*

Todas las demás capas son desarrolladas en forma neutral. En esta capa se tienen librerías para manejar sockets, tareas, u operaciones de criptografía que son específicas de la plataforma. Esto se hace para solo cambiar esta capa y no todas las demás al cambiar de plataforma.

2.4.2 SDK.

La primera parte del SDK trata de la implementación de los servicios de OPC UA, como son la creación de nodos, eventos y métodos. Además, se maneja el espacio de direcciones, atributos y propiedades en el SDK del servidor. También se maneja información de diagnóstico y de errores.

La segunda parte del SDK maneja el almacenamiento y validación de certificados de seguridad, la configuración de aplicaciones y el manejo de logs.

Actualmente existen SDK desarrollados en ANSI-C, C#, C++ y JAVA

2.4.3 Aplicación Cliente y Servidor.

Una aplicación cliente OPC UA generalmente tiene una interfaz de usuario que puede ser tan simple como un explorador del espacio de direcciones como un Sistema de

monitoreo Humano-Maquina (HMI). La aplicación cliente tiene un SDK de cliente y una configuración de acceso a nodos.

La aplicación servidor tiene dos partes; una parte que maneja el espacio de direcciones en memoria y otra que obtiene los datos de los sistemas de bajo nivel. El SDK del servidor es el encargado de leer y escribir datos al sistema de bajo nivel y de mantenerlos en memoria para un acceso rápido a ellos.

2.5 PATRONES DE ARQUITECTURA DE SISTEMAS OPC UA.

2.5.1 Patrones básicos de arquitectura.

2.5.1.1 *Cliente – Servidor OPC UA.*

Esta es la arquitectura más básica, en la que se tiene un cliente que emite peticiones y un servidor que responde con los datos tomados de los sistemas de bajo nivel como son los controladores industriales.

2.5.1.2 *Servidor encadenado OPC UA.*

Un servidor encadenado es aquel que tiene embebido un cliente OPC UA. Se utiliza si un cliente no soporta el perfil de un servidor, entonces se utiliza un servidor encadenado para proveer al cliente con el perfil deseado. Por ejemplo un cliente que solo tiene transporte HTTPs y quiere comunicarse con un servidor que solo implementa transporte OPC UA TCP. En la figura 2.26 se muestra un ejemplo de un servidor encadenado

.

2.5.1.3 *Comunicación Servidor a Servidor OPC UA.*

La comunicación servidor a servidor se la realiza con un cliente OPC UA embebido en cada servidor. Además, cada servidor puede atender a un cliente OPC UA externo. Esta arquitectura es utilizada cuando se requiere redundancia. En la figura 2.27 se tiene un bosquejo de una comunicación servidor a servidor.

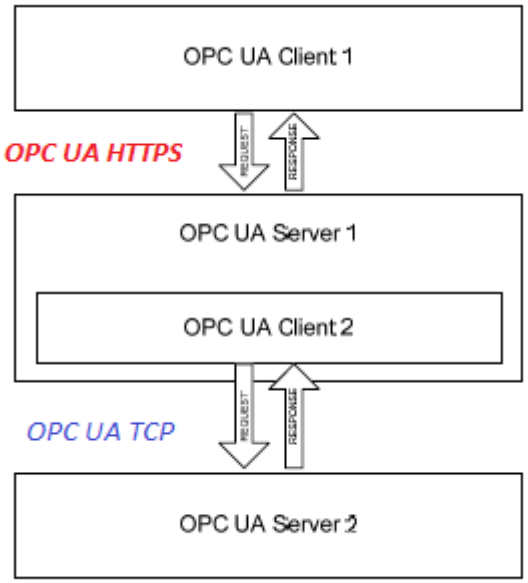

Figura 2.26: Servidor encadenado en OPC UA. Fuente: OPC Unified Architecture. Wolfgang Mahnke. 2009.

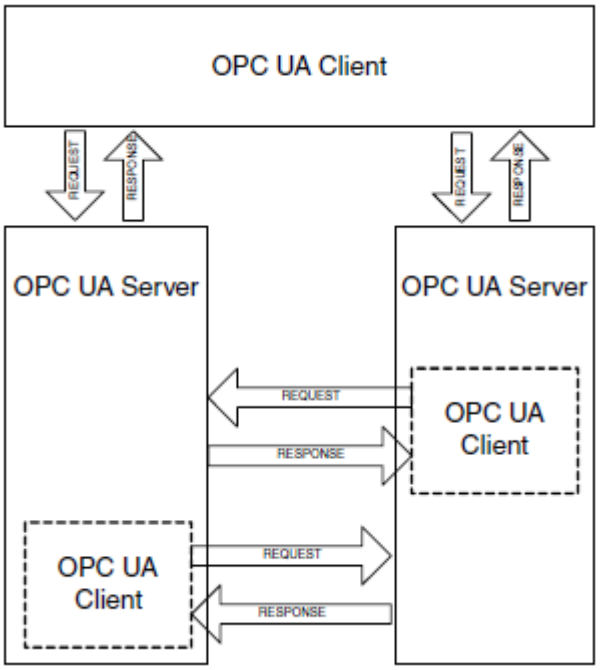

Figura 2.27: Comunicación servidor-servidor en OPC UA. Fuente: OPC Unified Architecture. Wolfgang Mahnke. 2009.

2.5.1.4 *Servidor de Agregación OPC UA.*

En este modelo se tiene un servidor con un cliente embebido que se conecta y almacena datos de varios servidores. Todos los datos son tratados por el servidor de agregación y mostrados hacia clientes externos. Uno de los ejemplos de utilización de este modelo es a nivel de producción, en donde se requieren datos de diferentes servidores los cuales manejan una parte del proceso, un producto puede pasar por varios procesos y cada proceso estar comandado por un controlador que expone sus datos por medio de un servidor OPC UA. En la siguiente figura 2.28 se muestra un diagrama de bloques de un servidor de agregación.

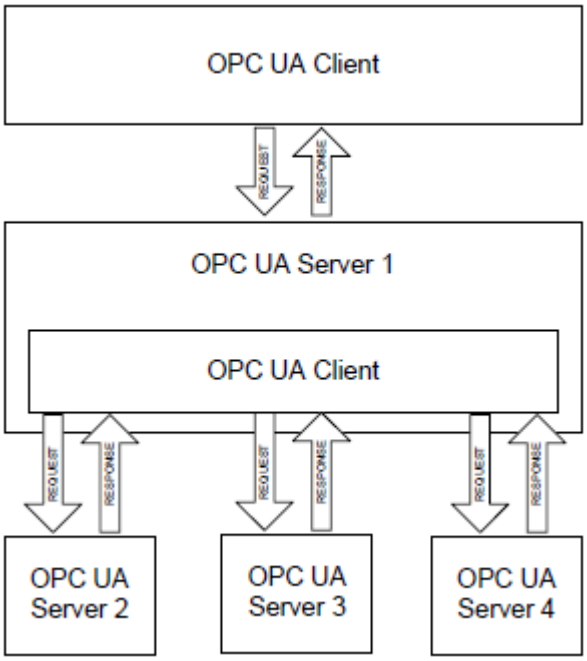

Figura 2.28: Servidor de agregación OPC UA. Fuente: OPC Unified Architecture. Wolfgang Mahnke. 2009.

2.5.2 Redundancia de clientes y Servidores.

Para aumentar la confiabilidad y tener sistemas críticos funcionando todo el tiempo se realiza la redundancia. La redundancia permite ejecutar una aplicación si otra del mismo tipo falla. Existen redundancia a nivel de clientes y redundancia a nivel de servidores.

2.5.2.1 *Redundancia de clientes OPC UA.*

Este modelo de redundancia es utilizado cuando se tiene sistemas de supervisión que no pueden detenerse por lo crítico del sistema. Se tiene un cliente principal y un cliente de respaldo. El servidor de respaldo esta todo el tiempo comprobando el estado del cliente principal y monitoreando los nodos del espacio de direcciones. Si existe una falla en el cliente principal, el estado de la sesión reporta al cliente de respaldo y transfiere las subscripciones hacia el cliente de respaldo. En la figura 2.29 se muestra un diagrama de bloques de la redundancia de clientes.

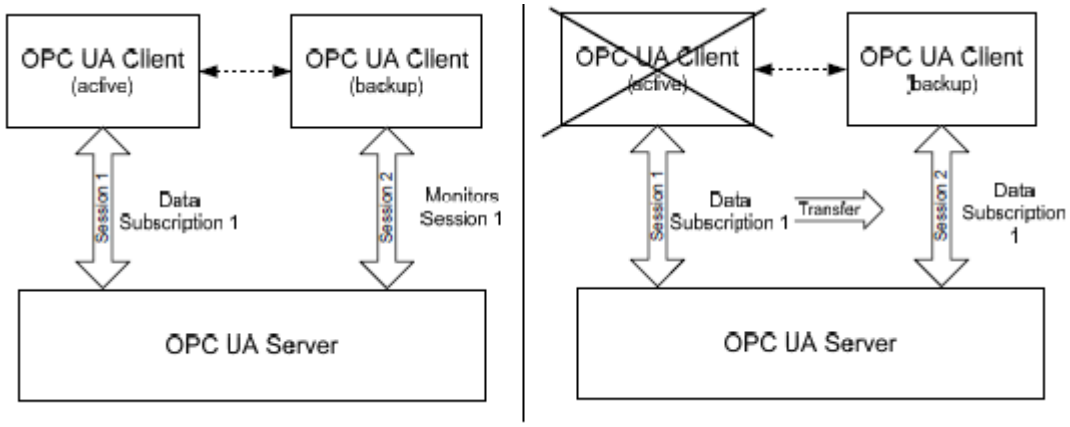

Figura 2.29: Redundancia en clientes OPC UA. Fuente: OPC Unified Architecture. Wolfgang Mahnke. 2009.

2.5.2.2 *Redundancia de servidores OPC UA.*

Existen dos tipos de redundancia de servidores OPC UA. La redundancia transparente en la que el cliente no se da cuenta de que hubo un error en un servidor y la redundancia no transparente en la que el cliente debe realizar un método para volver a conectarse al servidor. En la figura 2.30 se muestra un diagrama de bloques de la redundancia de servidores.

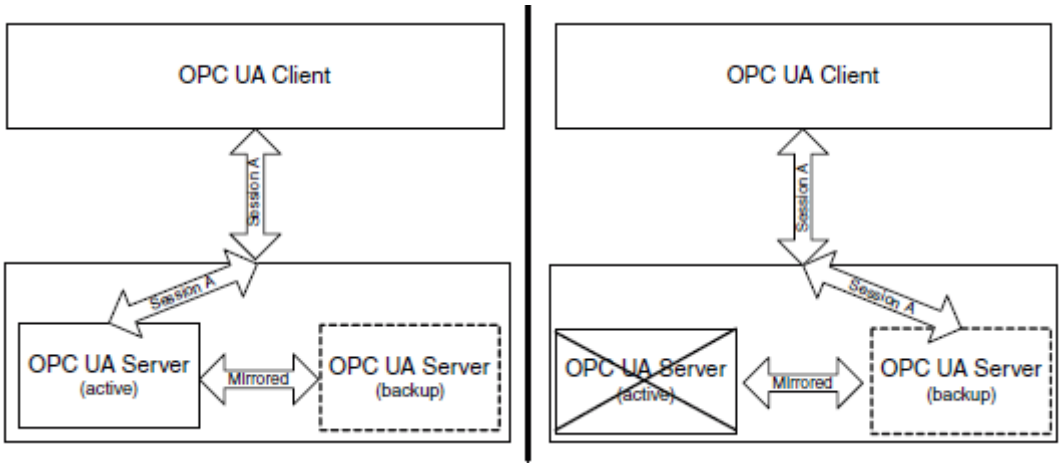

Figura 2.30: Redundancia transparente de servidores en OPC UA. Fuente: OPC Unified Architecture. Wolfgang Mahnke. 2009.

2.5.3 Descubrimiento de Servidores.

En ciertos ambientes de producción se tienen servidores con diferentes perfiles de seguridad y en diferentes segmentos de red, para ubicarlos se necesita de entidades de descubrimiento. Existen dos tipos de entidades de descubrimiento: el servidor de descubrimiento local (LDS) y el servidor de descubrimiento global (GDS).

El servidor de descubrimiento local reside en la misma máquina que el servidor OPC UA y expone los puntos finales de conexión de los múltiples servidores OPC UA que podrían correr en la máquina.

El servidor de descubrimiento global tiene información de los servidores en una red y debe conocerse su URL y su puerto. Este permite obtener los puntos finales de conexión de los diferentes servidores OPC UA que se encuentran en la red, no necesariamente en el mismo segmento. Si los servidores OPC UA están en distribuidos en una red muy extensa se debe instalar un servidor de descubrimiento global. Cada cliente debe tener registrado con anterioridad la dirección del servidor de descubrimiento global. Cuando el cliente desea obtener la dirección de un servidor OPC UA envía una petición con el servicio FindServers al servidor de descubrimiento global, el cual responde con la lista de las

direcciones de los servidores de descubrimiento local que tienen la información de los servidores OPC UA instalados.

2.5.4 Auditoria.

Los registros de auditoria (Logs de auditoria) en OPC UA guardan información de actividades y eventos producidos en la aplicación. Cada entrada de los registros de auditoria tiene un ID para identificación. Cada evento que se produzca genera una entrada en el registro de auditoria, como por ejemplo el establecimiento de un canal seguro o la activación de una sesión, o el cambio de una suscripción. Para cada servicio implementado en OPC UA y alguna actividad relacionada con ellos se tiene una entrada en el registro de auditoria. Con este registro de auditoria se puede observar los eventos sospechosos producidos en la comunicación OPC UA. En la figura 2.31 se muestra un ejemplo de una entrada de auditoria al activarse una sesión.

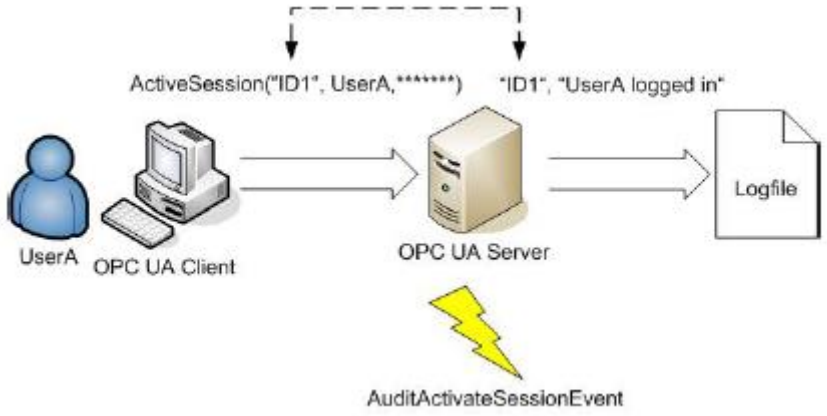

Figura 2.31: Auditoria de un evento al activar una sesión OPC UA. Fuente: OPC Unified Architecture. Wolfgang Mahnke. 2009.

3 CAPÍTULO 3: ANÁLISIS COMPARATIVO DE ALTERNATIVAS.

3.1 INTRODUCCIÓN AL ANALISIS COMPARATIVO MEDIANTE SIMULACIÓN.

En el presente capitulo se realizará una simulación entre un cliente y un servidor OPC UA ejecutándose en máquinas virtuales de dos diferentes computadoras portátiles, con datos de tipo doble en una cantidad de 5 a 1000 datos para observar el comportamiento del protocolo de comunicaciones y se capturaran paquetes de datos por medio del software Wireshark para contabilizar el ancho de banda utilizado entre el cliente y servidor.

Además, se utilizaran varios modos y políticas de seguridad para observar el manejo de estas opciones y realizar un análisis comparativo.

También se probará el cambio del tiempo de muestreo de la lectura de las variables para observar el cambio del uso del ancho de banda.

Por último se realizará una prueba con el cliente OPC UA corriendo en el sistema operativo Windows Server 2012 R2 y otra prueba en Linux Ubuntu 14.04 LTS para comprobar la independencia de la ejecución del protocolo en diferentes sistemas operativos. A continuación se describe el método y las herramientas utilizadas para el análisis comparativo mediante simulación.

3.2 EQUIPOS PARA LOS ESCENARIOS DE PRUEBA.

Para la realización de las pruebas se utilizaron equipos nuevos los cuales se configuraron para los escenarios de pruebas propuestos y utilizando una red LAN privada para aislar el tipo de flujo de datos a tener en prueba. En la siguiente Tabla 3.1 se muestran los equipos físicos para los escenarios de prueba.

Tabla 3.1: Equipos utilizados en los escenarios de prueba. Fuente autor.

Equipos	Uso
Laptop HP ProBook 4730s, con procesador intel Core i7 de 2.2 GHz, con 8 GB de memoria RAM y sistema operativo Windows 10 de 64 bits. Instalado el software para manejo de máquinas virtuales "VirtualBox" versión 5.0.20	Computador portátil para la simulación del servidor OPC UA seleccionado y del Sistema Scada.
Laptop Dell Inspiron M531R-5535 con procesador AMD A8 de 1.7 GHz, con 6 GB de memoria RAM y sistema operativo Windows 10 de 64 bits. instalado el software para manejo de máquinas virtuales "VirtualBox" versión 5.0.20	Computador portátil para la simulación del cliente OPC UA seleccionado y del Controlador industrial basado en PC.
Switch de datos TP-LINK TL-SG105E de 5 puertos de red Gigabit Ethernet.	Switch de datos con funcionalidad de port mirroring para el monitoreo de tráfico de datos.
Red LAN privada	Red lan Privada para la comunicación entre los computadores portátiles de prueba.

En cada uno de los computadores portátiles de los escenarios de prueba se instalaron y configuraron máquinas virtuales para los diferentes escenarios de prueba. En la siguiente Tabla 3.2 se muestran las máquinas virtuales creadas y configuradas.

Tabla 3.2: Máquinas virtuales utilizadas en los escenarios de prueba. Fuente autor.

Máquinas virtuales en Laptop HP Probook 4730s	Uso
Máquina virtual con 2 microprocesadores, 2 GB de memoria y 60 GB de disco duro virtual, con sistema operativo Windows Server 2012 R2 a 64 bits.	Máquina virtual utilizada para la ejecución del servidor OPC UA seleccionado.
Máquina virtual con 2 microprocesadores, 2 GB de memoria y 60 GB de disco duro virtual, con sistema operativo Windows Server 2008 R2 a 64 bits.	Máquina virtual utilizada para la ejecución del sistema SCADA seleccionado.
Máquinas virtuales en Laptop DELL Inspiron M531R	**Uso**
Máquina virtual con 2 microprocesadores, 2 GB de memoria y 60 GB de disco duro virtual, con sistema operativo Windows Server 2012 R2 a 64 bits.	Máquina virtual utilizada para la ejecución del cliente OPC UA seleccionado en sistema operativo Windows.
Máquina virtual con 2 microprocesadores, 2 GB de memoria y 60 GB de disco duro virtual, con sistema operativo Linux versión Ubuntu 14.04 LTS a 64 bits.	Máquina virtual utilizada para la ejecución del cliente OPC UA seleccionado en sistema operativo Linux.
Máquina virtual con 2 microprocesadores, 2 GB de memoria y 60 GB de disco duro virtual, con sistema operativo Windows 7 a 32 bits.	Máquina virtual utilizada para la ejecución del controlador industrial basado en PC.

3.3 CONFIGURACION GENERAL DE LA RED DE PRUEBA.

En la siguiente figura 3.1 se muestra la configuración de la red LAN de prueba. Para el monitoreo del tráfico de datos desde y hacia el servidor y cliente OPC UA se configuró al switch de datos TP-LINK TL-SG105E para que replique el tráfico de datos que entra y sale de los primeros 4 puertos Gigabit Ethernet hacia el puerto 5.

El puerto 5 del switch de datos se conecta a una de las computadores portátiles en donde se encuentra el software analizador de protocolos de red “Wireshark” para realizar la toma de captura de datos de comunicación OPC UA sobre el protocolo TCP.

La dirección de red utilizada es la de una red LAN privada con dirección de red 192.168.0.x y mascara de subred 255.255.255.0. En la figura 3.1 se muestran las asignaciones de direcciones IP y mascara de subred de los computadores portátiles.

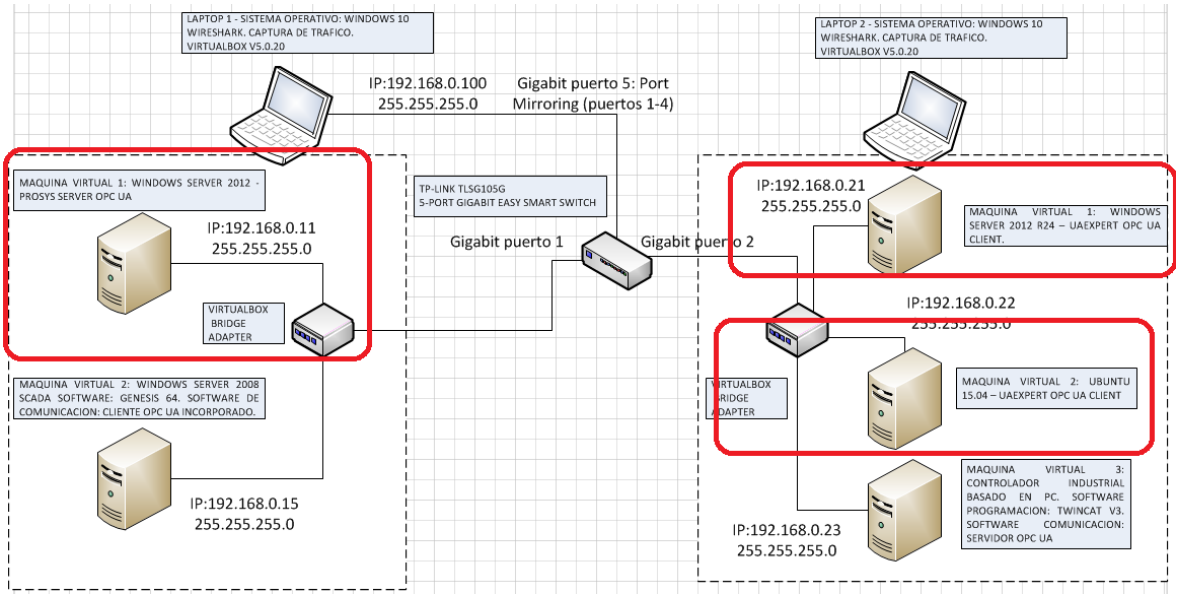

Figura 3.1: Esquema de la red LAN para los escenarios de prueba. Fuente el autor.

Para la configuración del switch de datos, se instaló el software "Easy Smart Configuration Utility" sobre uno de los computadores portátiles de prueba y se conectó el equipo de prueba con el switch de datos. Al momento de terminar con la instalación del software y ejecutarlo, se realiza un rastreo de los dispositivos disponibles sobre la red de prueba. Al encontrar el switch de datos se ingresa a su configuración mediante el usuario y contraseña de defecto. En la siguiente figura 3.2 se muestra la página de rastreo del software de configuración del switch de datos.

Uno de los computadores portátiles de prueba tiene dos interfaces de red; una propia del computador portátil y otra por medio de un convertidor USB 3.0 a LAN (RJ45). Las máquinas virtuales se conectan por medio del adaptador USB a LAN en una configuración de puente entre la interfaz virtual y la interfaz del adaptador USB, y la interfaz de red propia del computador se la utiliza con el software Wireshark para capturar los datos del puerto 5 del switch de datos. En la siguiente figura 3.2 se muestra la configuración del switch de datos para la función de "Port Mirroring" de los puertos 1 al 4 hacia el puerto 5.

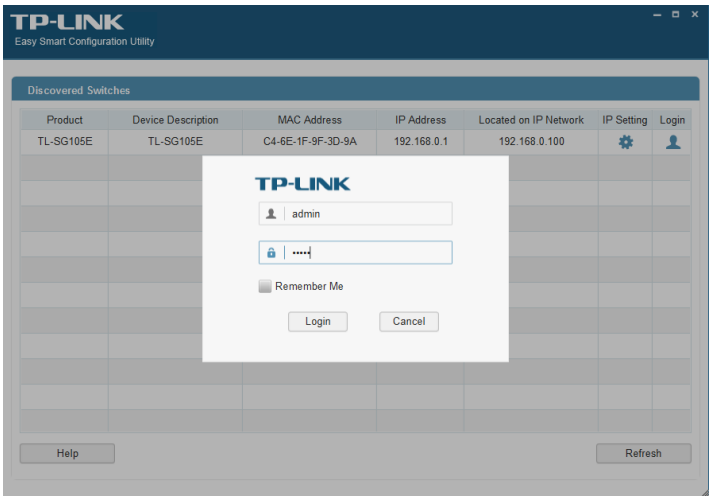

Figura 3.2: Página inicial de ingreso al switch de datos TP-LINK TL-SG105E. Fuente el autor.

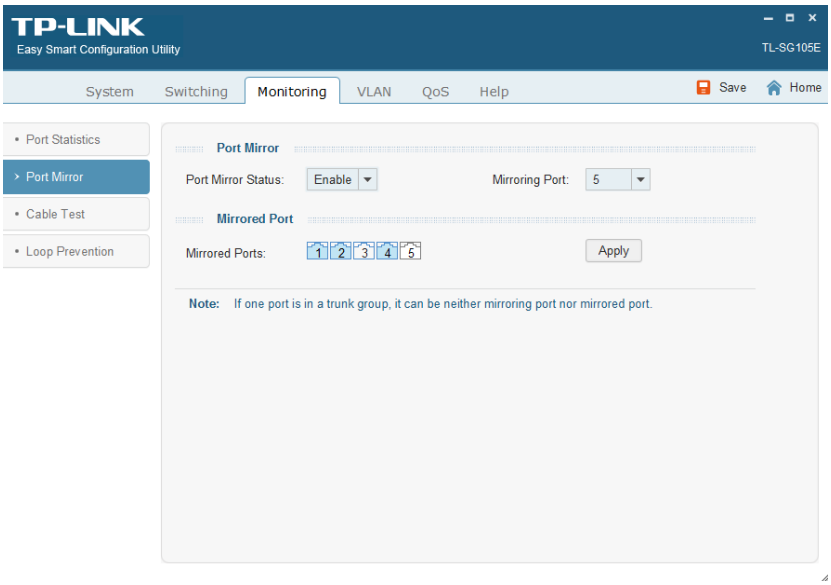

Figura 3.3: Configuración de "Port Mirroring" de los puertos 1, 2 y 4 hacia el puerto 5 del switch de datos TP-LINK TL-SG105E. Fuente autor.

3.4 ALTERNATIVAS DE HERRAMIENTAS PARA CLIENTES OPC UA.

3.4.1 Cliente OPC UA - Prosys

El Cliente OPC UA de la empresa de software Prosys, es un cliente desarrollado con el SDK OPC UA en Java. Puede ser ejecutado en el sistema operativo Windows, Linux o MAC. Es un cliente grafico el cual indica el espacio de direcciones de un servidor OPC UA en forma de árbol y es fácil su navegación. Permite el monitoreo de variables OPC mediante subscripciones y métodos de lectura y escritura. En la figura 3.4 se muestra la ejecución del cliente OPC UA Prosys sobre una de las máquinas virtuales con el sistema operativo Windows Server 2012 R2 64 bits.

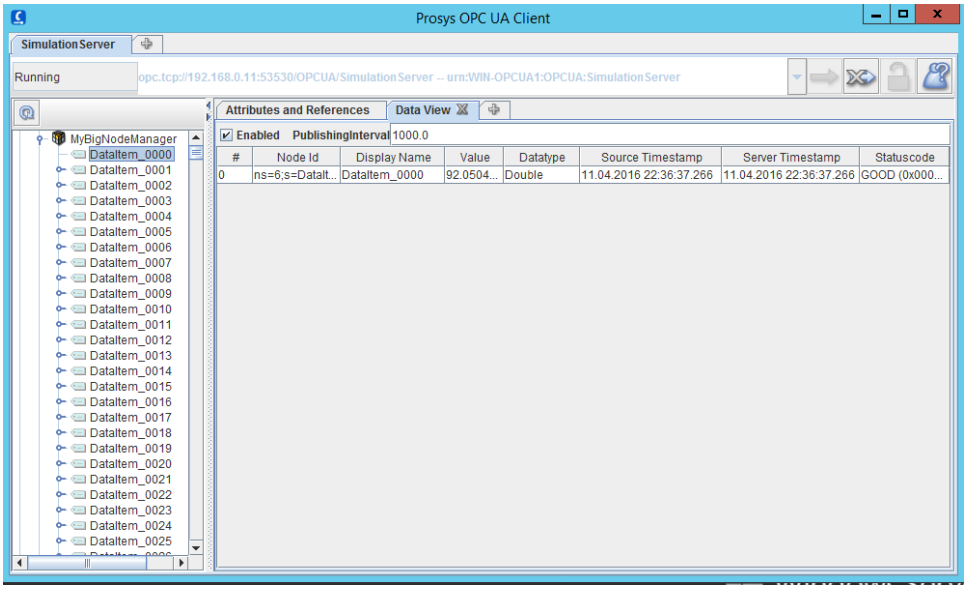

Figura 3.4: Cliente OPC UA de Prosys ejecutándose sobre una máquina virtual con Windows Server 2012. Fuente autor.

3.4.2 Cliente OPC UA - UaExpert.

EL Cliente OPC UA de la empresa de software Unified Automation, es un cliente desarrollado sobre el SDK/toolkit en C++ para clientes OPC UA. Este cliente puede correr sobre el sistema operativo Windows o Linux. Permite navegar de forma gráfica sobre el espacio de direcciones de un servidor OPC UA. La interfaz gráfica del cliente posee menús sobre los cuales se puede configurar una vista OPC UA DA para las subscripciones de

variables OPC UA, y además, permite la escritura de valores sobre las variables. En la figura 3.5 se muestra la ejecución del cliente OPC UA UAExpert sobre una de las máquinas virtuales con el sistema operativo Windows Server 2012 R2 64 bits.

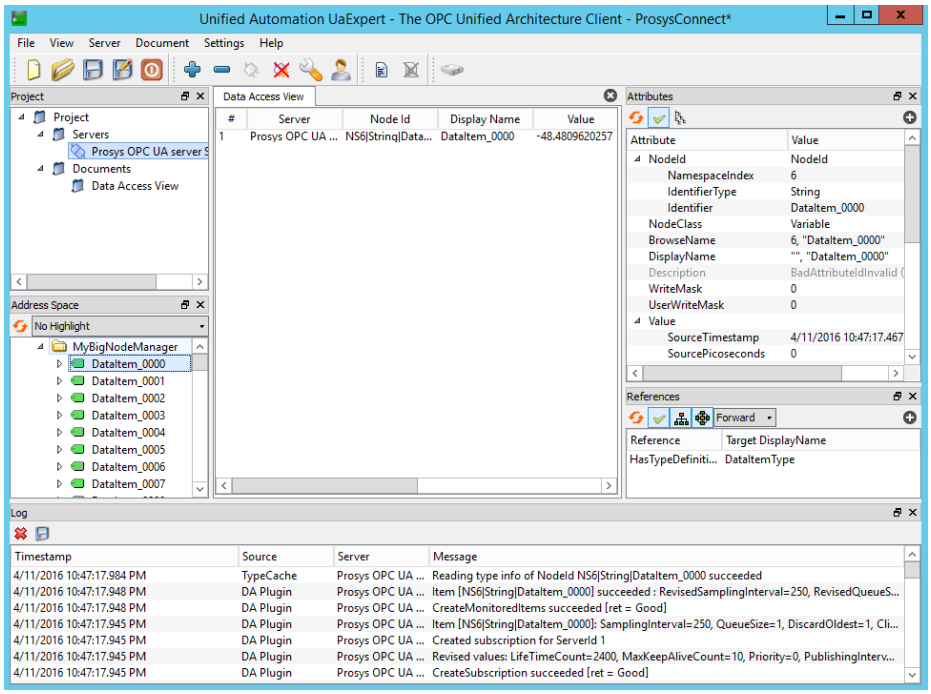

Figura 3.5: Cliente OPC UA de Unified Automation ejecutándose sobre una máquina virtual con Windows Server 2012 R2. Fuente autor.

3.4.3 Cliente OPC UA – OPC Foundation.

OPC Foundation en su portal web ofrece a los usuarios registrados la posibilidad de descargarse un cliente y un servidor OPC UA de prueba con capacidades muy reducidas para probar las capacidades del protocolo OPC UA. En la figura 3.6 se muestra a la aplicación Cliente OPC UA de OPC Foundation corriendo sobre el sistema operativo Windows.

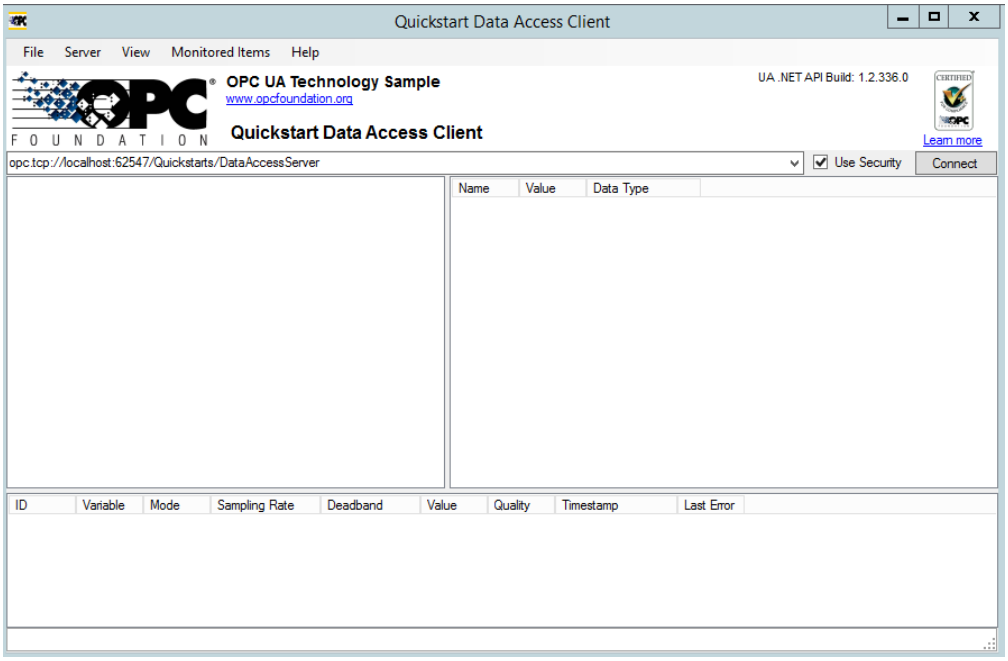

Figura 3.6: Cliente OPC UA de OPC Automation ejecutándose sobre una máquina virtual con Windows Server 2012 R2. Fuente el autor.

3.4.4 Cuadro comparativo de alternativas de clientes OPC UA.

En la siguiente tabla se observa un analisis de los 3 clientes OPC UA mencionados.

Tabla 3.3: Cuadro comparativo de capacidades de clientes OPC UA analizados. Fuente el autor.

Cliente OPC	Sistema Operativo soportado	Exploración de datos	Interfaz gráfica	Lenguaje SDK	Configuración del tiempo de muestreo de las variables OPC	Cambio del número de variables OPC leídas
Prosys	Windows, Linux, MAC OS	Datos básicos, Estructuras	Si	Java	Fijo (1000 ms)	SI
UaExpert	Windows, Linux	Datos básicos, Estructuras	Si	C++	Variable (desde 100 ms)	Si
OPC Foundation	Windows	Datos básicos, Estructuras	Si	.NET	Variable (desde 1000 a 5000 ms)	Limitado número

3.5 ALTERNATIVAS DE HERRAMIENTAS PARA SERVIDORES OPC UA.

3.5.1 Servidor de simulación OPC UA – Prosys

El servidor de simulación OPC UA de la empresa Prosys, permite la conexión de clientes por medio de los modos de seguridad y políticas de seguridad establecidos en el estándar OPC UA. Además, provee de pestañas con funcionalidades que permiten observar el espacio de direcciones y las sesiones activas con el servidor. También tiene funcionalidades para cambiar el tiempo se simulación y provee de un nodo del espacio de direcciones con 100 elementos del tipo doble con datos simulados que cambian con el tiempo de simulación que puede ser establecido a un valor deseado. En la figura 3.7 se muestra la ejecución del servidor OPC UA de Prosys.

Figura 3.7: Servidor de Simulación OPC UA de Prosys ejecutándose sobre una máquina virtual con Windows Server 2012 R2. Fuente autor.

3.5.2 Servidor demo OPC UA – Unified Automation.

El servidor de demo de la empresa Unified Automation viene en dos versiones una desarrollada sobre el SDK basado en ANSI C y la otra versión desarrollada con el SDK basado en C++.

El servidor demo no posee una interfaz gráfica de fácil manejo e inclusive solo muestra una pantalla de consola que indica el estado del servidor. En la figura 3.8 se

muestra la ejecución del servidor demo OPC UA de Unified Automation desarrollado en el SDK de ANSI C.

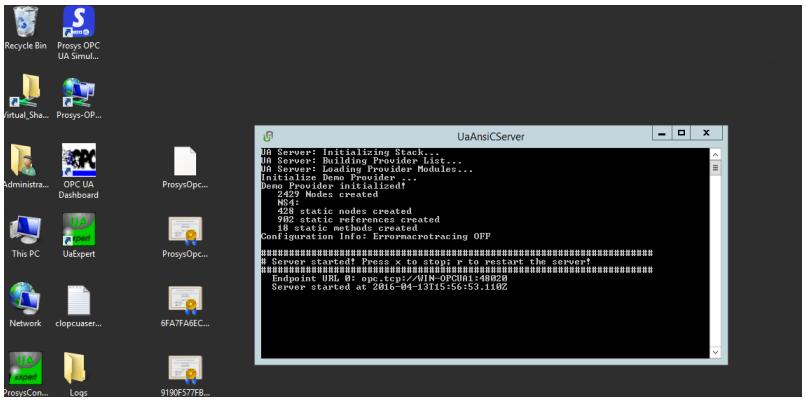

Figura 3.8: Servidor de Simulación OPC UA de Unified Automation basado en ANSI C ejecutándose sobre Windows Server 2012 R2. Fuente autor.

La versión del servidor demo de "Unified Automation" desarrollada en el SDK basado en C++, posee únicamente una interfaz de consola en la cual solo se puede observar el estado del servidor. A diferencia del servidor ANSI C, el servidor OPC UA basado en C++ pose una herramienta de administración en la cual se puede configurar cuáles métodos y políticas de seguridad permitirá el servidor utilizar para la conexión con un cliente. Además, permite ver y autorizar el intercambio de los certificados de instancia de aplicación entre el servidor y el cliente. En la figura 3.9 se muestra la ejecución del servidor OPC UA basado en C++.

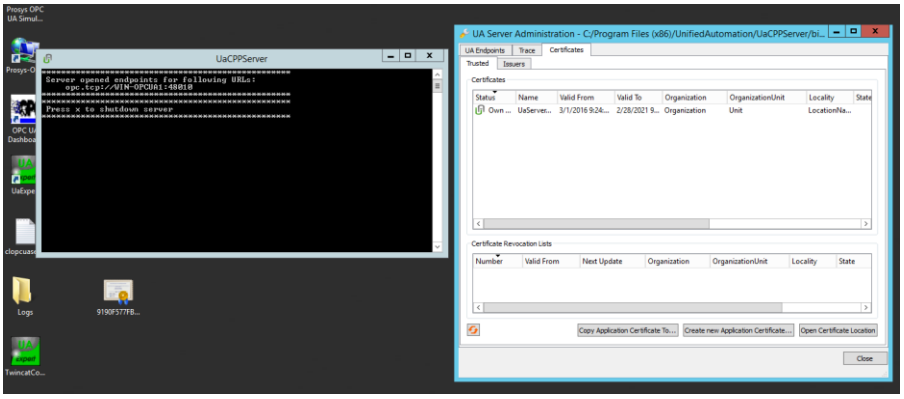

Figura 3.9: Servidor de Simulación OPC UA de Unified Automation basado en C++ ejecutándose sobre Windows Server 2012 R2. Fuente autor.

3.5.3 Servidor demo OPC UA – OPC Foundation.

Dentro de la aplicación de demo de OPC Foundation se puede ejecutar a un servidor OPC UA. La interfaz del servidor OPC UA muestra las sesiones activas y las subscripciones a las variables que son accesadas por el cliente. Como la aplicación de OPC Foundation es para prueba solo tiene un número limitado de nodos y variables a acceder. En la figura 3.9 se muestra la ejecución del servidor OPC UA de demo proveída por OPC Foundation.

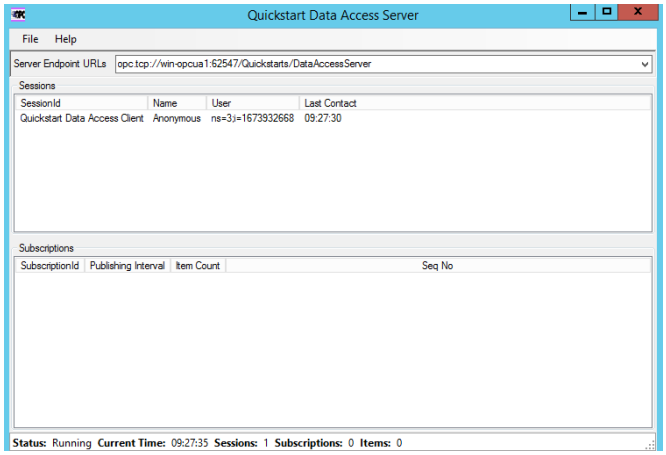

Figura 3.10: Servidor de demo OPC UA desarrollado por OPC Foundation ejecutándose sobre Windows Server 2012 R2. Fuente autor.

3.5.4 Cuadro comparativo de alternativas de servidores OPC UA.

En la siguiente tabla se observa las ventajas y desventajas de los 4 servidores OPC UA mencionados.

Tabla 3.4: Cuadro comparativo de capacidades de servidores OPC UA analizados. Fuente el autor.

Servidor OPC	Sistema Operativo soportado	Presentación de datos	Interfaz gráfica	Lenguaje SDK	Visor y Configuración de modos y políticas de seguridad	Visor de eventos de conexión entre servidor y cliente	Cambio del tiempo de publicación de variables OPC	Herramienta de configuración de permisos de usuarios
Prosys	Windows, Linux, MAC OS	Datos Básicos, Estructuras	Si	Java	Visor - Configuración	SI (detallado)	Si	Si
Unified Automation ANSI C	Windows, Linux, MAC OS	Datos Básicos, Estructuras	No	ANSI C	No	No	No	No
Unified Automation C++	Windows, MAC OS	Datos básicos, Estructuras	No	C++	Visor - Configuración	No	No	Si
OPC Foundation	Windows	Datos básicos, Estructuras	Si	.NET	Visor - Configuración	Si (básico)	No	No

3.6 ELECCIÓN DEL SOFTWARE A UTILIZAR PARA LA SIMULACIÓN DE ALTERNATIVAS.

Se seleccionó como cliente OPC UA al cliente desarrollado por Unified Automation, debido a que este posee una interfaz gráfica más completa e inclusive tiene herramientas de monitoreo que permiten añadir fácilmente las variables OPC UA desde el explorador de espacio de direcciones hacia las vista de subscripciones de la interfaz gráfica. Además, puede ser ejecutado en el sistema operativo tanto con Windows como en Linux.

El servidor OPC UA seleccionado para la simulación es el desarrollado por Prosys, debido a que posee una interfaz gráfica con varias pestañas que muestran las sesiones y permiten el monitoreo de usuarios y un log de las acciones realizadas por los clientes. Además, en su espacio de direcciones posee un nodo con 1000 variables del tipo doble que se pueden monitorear y poder observar el tiempo y ancho de banda utilizado por un determinado número de variables hasta 1000 variables. También, se puede cambiar el tiempo de simulación hasta en 100 ms y así poder observar el ancho de banda y el tiempo de respuesta al cambiar el tiempo de muestreo del cliente hacia el servidor. Estas características eran vitales para el tipo de análisis requerido.

3.7 DESCRIPCIÓN DEL ESCENARIO DE PRUEBA DE SIMULACIÓN.

Las pruebas se realizaron con un número determinado de variables del tipo doble, este tipo de datos se seleccionó porque es el más usado para la lectura de variables de proceso con una precisión de 32 bits, como por ejemplo la lectura de la temperatura de un material metálico con precisión de centésimas de grado centígrado, la velocidad del motor en centésimas de revoluciones por minuto o la presión o flujo de agua en mililitros por segundo. Además, se pudo realizar la lectura de datos más elaborados como estructuras de datos, pero esto conlleva a particularizar la lectura para una determinada aplicación como por ejemplo en el caso de centrales hidroeléctricas se podría tener una estructura del tipo generador y tener algunas variables como el voltaje, potencia y corriente como tipos de datos dobles.

Se tomaron capturas de paquetes de la lectura de esas variables con un tiempo de muestreo determinado. Las pruebas inician con ningún modo de seguridad, es decir los datos no están firmados ni encriptados, y posteriormente se prueban los demás modos de seguridad y políticas de seguridad para observar el comportamiento en el consumo de ancho de banda. Las pruebas comienzan con la lectura y suscripción de 5 variables a un (1) segundo de tiempo de adquisición. Posteriormente se aumentan el número de variables de 5, 10, 20, 50, 100, 200 y 1000 variables dobles para observar el comportamiento al aumentar el número de variables a un determinado tiempo de adquisición. También, se realizan pruebas con un tiempo de adquisición de 1 segundo, 500 milisegundos y 100 milisegundos, este último sería el tiempo de adquisición más rápido. Adicionalmente, se prueba con el cliente OPC UA UaExpert ejecutándose en una máquina virtual con el sistema operativo Windows Server 2012 R2 y otra prueba ejecutándose sobre Ubuntu 14.04 LTS (Long Term Support) a 64 bits para comparar y ver alguna diferencia en la adquisición ejecutándose en diferentes sistemas operativos. Por último, se realiza una adquisición con un modo de autenticación anónimo, con un usuario - contraseña y con certificado de usuario.

Al abrir la conexión entre servidor y cliente OPC UA con ningún modo de seguridad y ninguna política de seguridad, el servidor y cliente no intercambian certificados de instancia de aplicación que valida la autenticidad de las aplicaciones que están incluidas en la comunicación OPC UA. Si se realiza con un modo de seguridad firmado o firmado-encriptado, se debe intercambiar los certificados de aplicación del cliente con el servidor y viceversa, para esto se puede mover manualmente el certificado de la aplicación del cliente y copiarlo en la máquina que se esté ejecutando la aplicación del servidor OPC. Tanto el servidor como el cliente poseen carpetas de archivos para la funcionalidad de la administración de los certificados. Para que los certificados sean considerados confiables, estos deben ser intercambiados y copiados en la carpeta de manejo de certificados y dentro de la subcarpeta de certificados confiables (Trust). Adicionalmente del modo manual de intercambio de certificados, las aplicaciones que se van a utilizar en las pruebas poseen la capacidad de pedir e intercambiar certificados sobre la conexión y preguntar en la interfaz de usuario si el certificado de la aplicación que está intentando conectarse es confiable o no. En la figura 3.11 se muestra los certificados tanto de la aplicación de cliente OPC UA UaExpert que ha sido considerada confiable por el servidor OPC UA Prosys.

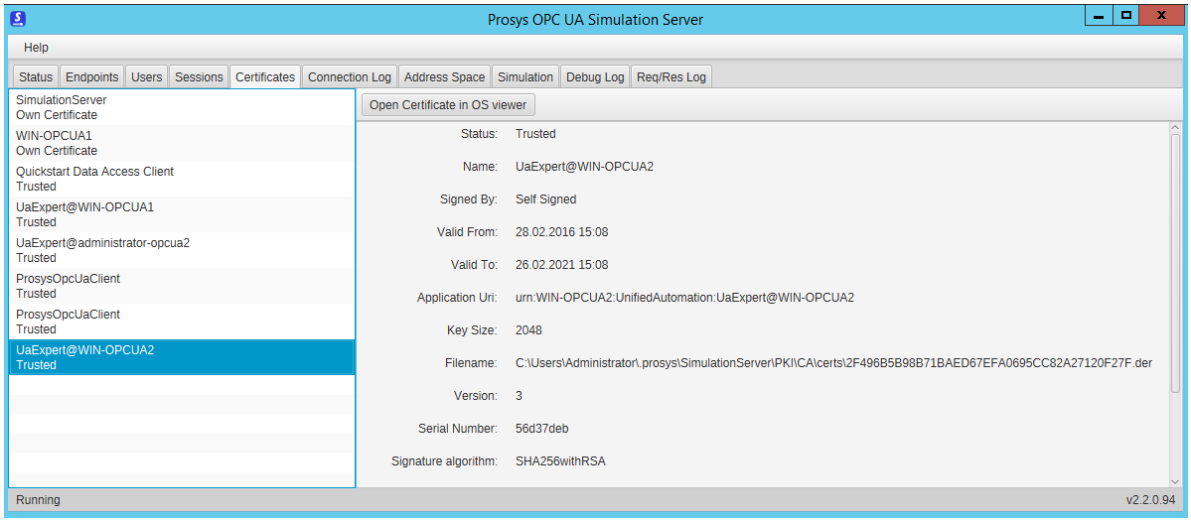

Figura 3.11: Certificado del cliente OPC UA UaExpert considerado confiable por el servidor OPC UA Prosys. Fuente autor.

3.7.1 Escenario de prueba 1.

En este escenario de prueba se realiza una conexión **sin modo de seguridad, usuario anónimo y tiempo de muestreo de 1 segundo.** Se realiza una conexión real entre los computadores portátiles de prueba ejecutándose máquinas virtuales con el sistema operativo Windows Server 2012 R2. La máquina virtual en la cual está ejecutándose el servidor OPC UA Prosys se configuró con la dirección IP: 192.168.0.11 y el puerto que utiliza para aceptar las conexiones OPC UA es el 53530, como se muestra en la figura 3.12.

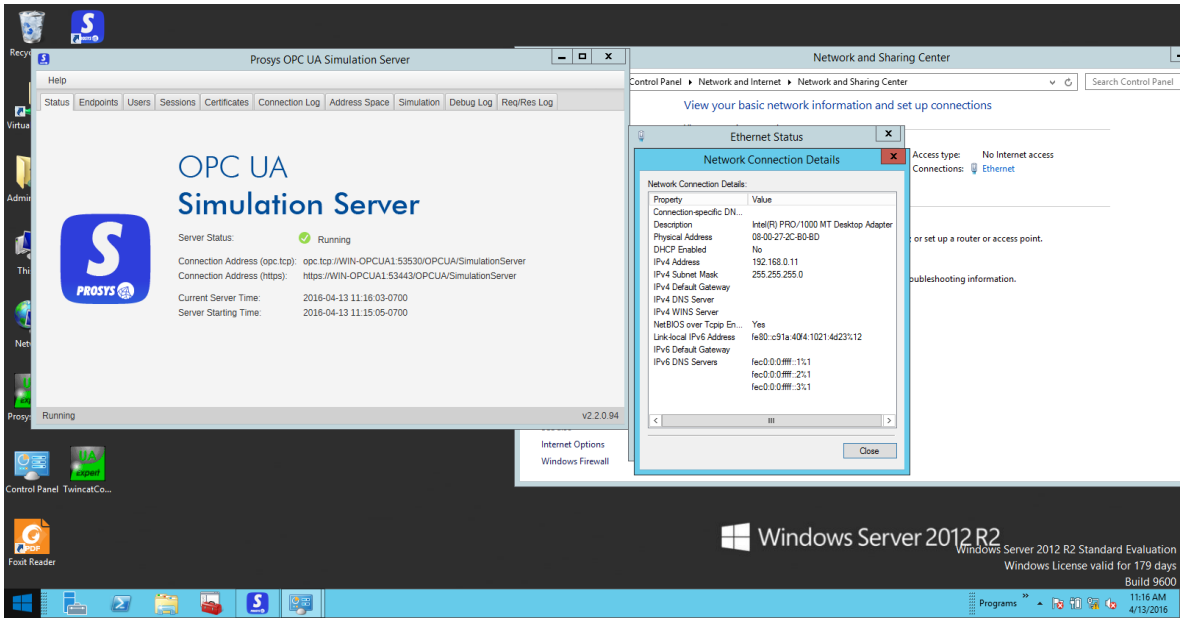

Figura 3.12: Configuración de dirección IP de la máquina virtual servidor OPC UA Prosys.
Fuente el autor.

La máquina virtual en la cual está ejecutándose el cliente OPC UA UaExpert se configuró con la dirección IP: 192.168.0.21, como se muestra en la figura 3.13.

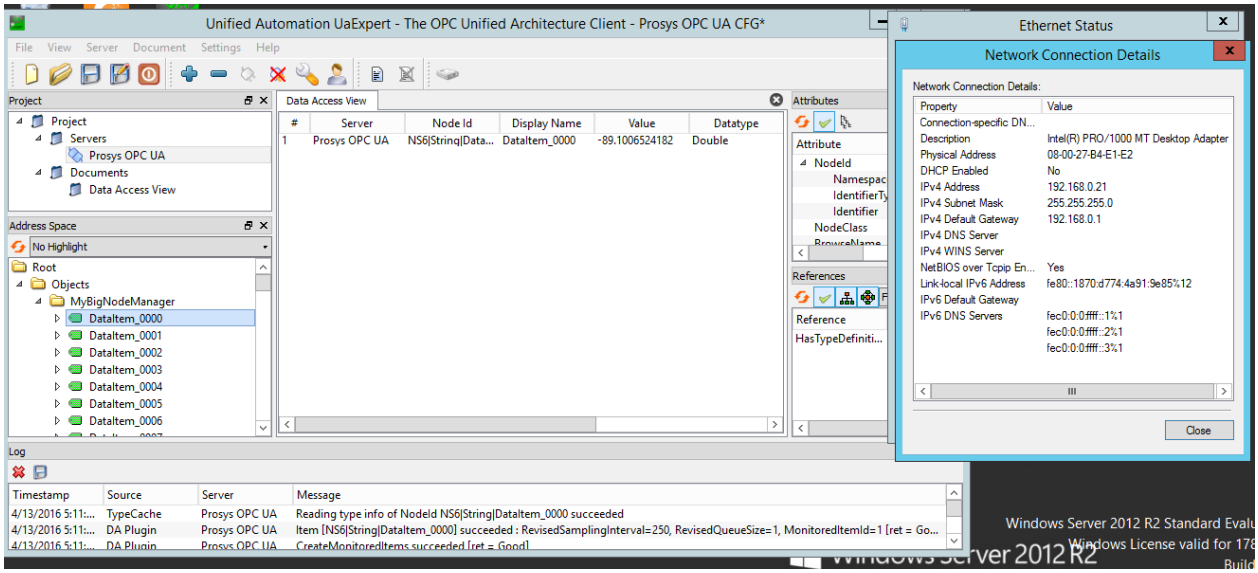

Figura 3.13: Configuración de dirección IP de la máquina virtual cliente OPC UA UaExpert. Fuente el autor.

La configuración del cliente OPC UA debe tener el URL del punto final del servidor OPC UA en este caso es "opc.tcp://WIN/OPCUA1:53530/OPCUA/SimulationServer", además, en la configuracion de seguridad debe estar seleccionada que no se va a usar ningún método y política de seguridad y el método de autenticación es anónimo. La configuración del cliente OPC UA se puede ver en la figura 3.14.

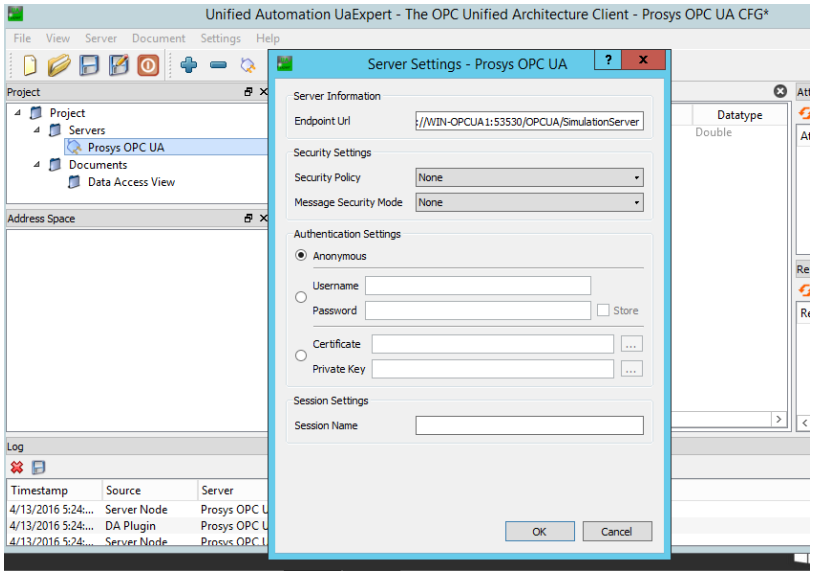

Figura 3.14: Configuración del URL del punto final del servidor OPC UA y la configuración de seguridad en el cliente OPC UA. Fuente autor.

Por último, debe configurarse al cliente OPC UA con tiempo de publicación de la suscripción a eventos de variables de 1000 milisegundos para que la lectura de variables sea a 1 segundo, como se muestra en la figura 3.15

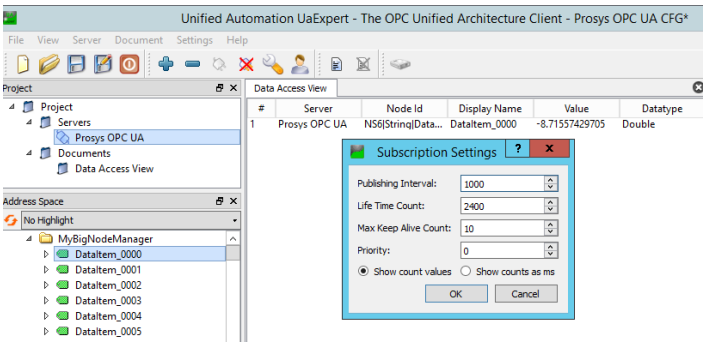

Figura 3.15: Configuración del intervalo de publicación en la configuración de la suscripción en el cliente OPC UA. Fuente autor.

3.7.1.1 *Escenario de prueba 1a (Lectura de 5 datos).*

Para el escenario de prueba se establece la conexión entre el cliente y servidor OPC UA y se capturan los paquetes OPC UA con el software Wireshark, a continuación se muestra una figura de los datos capturados.

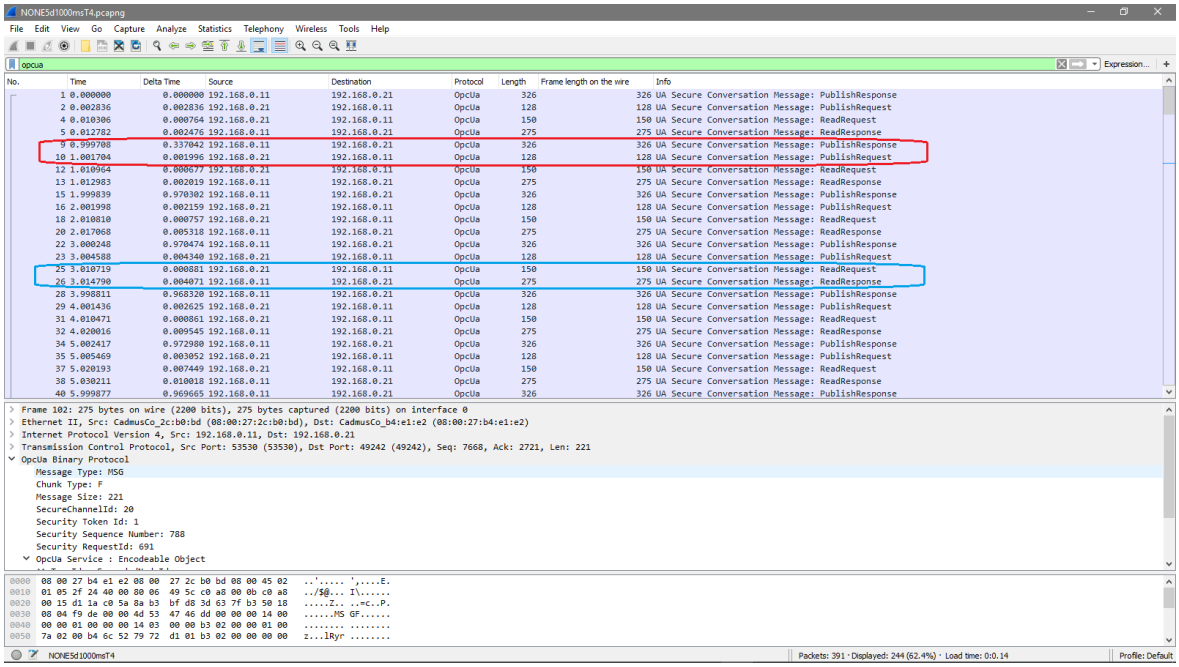

Figura 3.16: Captura del tráfico OPC UA. Escenario de prueba 1a. Fuente autor.

En la figura anterior se puede observar que existen dos tipos de conversaciones OPC UA, una es la conversación de petición de publicación de variables con su respuesta, y la otra conversación es la petición de lectura del estado del servidor y su respuesta.

En el caso de petición de la respuesta de publicación de variables se tiene una trama de 326 bytes y en su petición se tiene una trama con longitud de 128 bytes. En el caso de la petición de estado del servidor se tiene una trama de 150 bytes y su respuesta de 275 bytes.

En la siguiente figura 3.17 se puede observar un gráfico del uso del ancho de banda por la petición y respuesta de las subscripciones provenientes de las variables OPC UA.

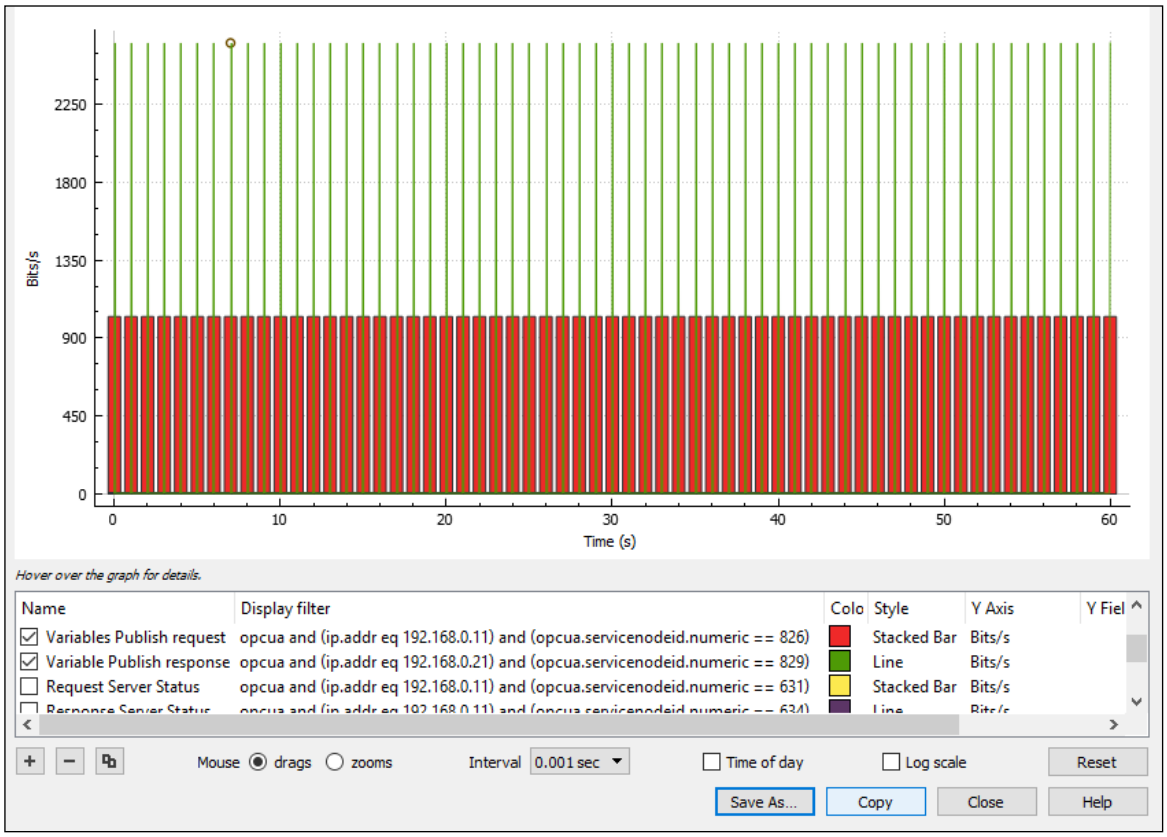

Figura 3.17: Uso del ancho de banda del tráfico OPC UA generado por las subscripciones. Escenario de prueba 1a. Fuente autor.

En el grafico anterior se puede observar que el ancho de banda utilizado es de 326x8=2608 bits y como la respuesta de la petición de lectura de datos es cada segundo se tiene 2608 bits/s, y de igual forma para la petición de las subscripciones con longitud de la trama de 128x8=1024 bits/s.

En la siguiente figura 3.18 se puede observar un gráfico del uso del ancho de banda por la petición y respuesta de la consulta del estado del servidor OPC UA.

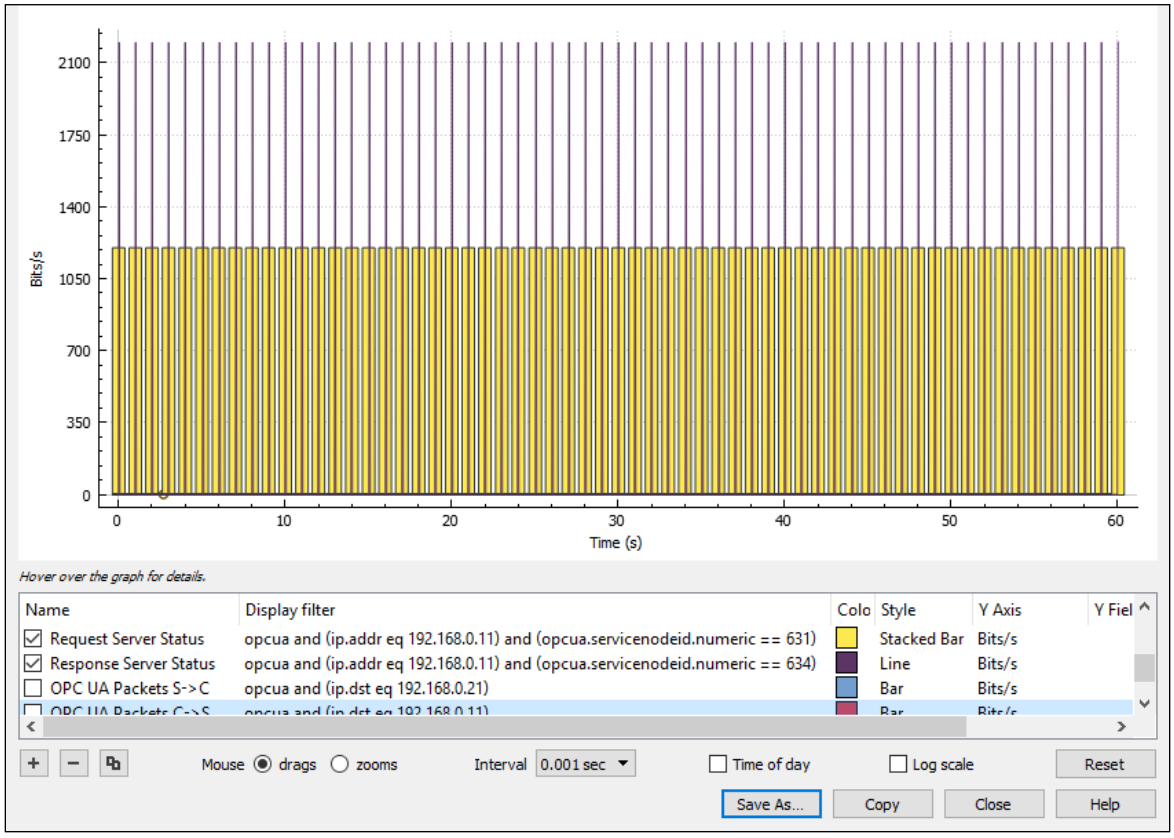

Figura 3.18: Uso del ancho de banda del tráfico OPC UA generado por las consultas del estado del servidor. Escenario de prueba 1a. Fuente autor.

En el gráfico anterior se puede observar que el ancho de banda utilizado es de 275x8=2200 bits/s, y de igual forma para la petición del estado del servidor se tiene una longitud de la trama de 150 bytes con un ancho de banda de 150x8=1200 bits/s.

Debido a los dos tipos de conversación OPC UA, y que cada paquete de petición y respuesta de las variables y del estado del servidor pueden ocurrir durante el lapso de 1 segundo, se tiene un gráfico resultante como el que se muestra en la figura 3.19.

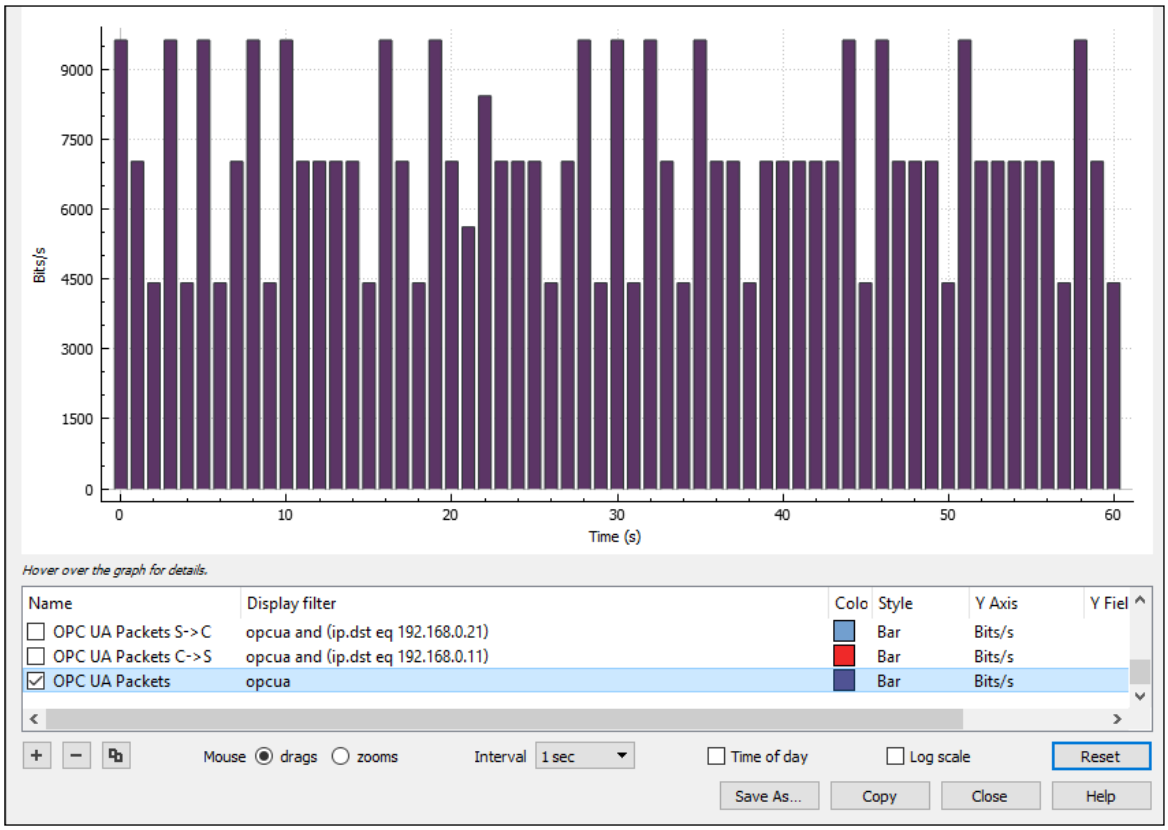

Figura 3.19: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 1a. Fuente autor.

En el gráfico anterior se puede observar que el ancho de banda puede llegar hasta más de 9000 bits/s con un promedio de 7146 bits/s, como se puede apreciar en la siguiente figura 3.20.

Statistics

Measurement	Captured	Displayed	Marked
Packets	391	244 (62.4%)	N/A
Time span, s	60.030	60.022	N/A
Average pps	6.5	4.1	N/A
Average packet size, B	167.5	219.5	N/A
Bytes	65360	53619 (82.0%)	0
Average bytes/s	1088	893	N/A
Average bits/s	8710	7146	N/A

Figura 3.20: Estadísticas de la conversación OPC UA. Escenario de prueba 1a. Fuente autor.

3.7.1.2 *Escenario de prueba 1b (Lectura de 11 datos).*

Al realizar la comparación con respecto a la prueba anterior de transmisión de 5 datos se puede observar que las cabeceras de Ethernet, IP y TCP/IP no cambian de tamaño y las cabeceras del propio paquete de OPC UA no cambian, lo que si cambia es el tamaño del cuerpo de mensaje a enviar debido a el nuevo tamaño de los mensajes que en la prueba actual es de 11 datos tipo doble.

Al realizar una inspección al cuerpo del mensaje OPC UA se puede observar que al enviar un dato doble se utiliza 30 bytes, de los cuales 4 bytes se utiliza para un identificador de manejo del cliente OPC UA para identificar a los datos y de 26 bytes relativos al valor del dato, De los 26 bytes relativos al valor del dato se tiene que se utiliza 1 byte en el campo "EncodingMask" que indica que campos se envía junto al valor del dato, que en el presente caso son los sellos de tiempo de la fuente de petición y del servidor. Para el valor del dato en si se utiliza 9 bytes; 1 byte indica el tipo de dato y los restantes 8 bytes para representar el valor de tipo de dato doble. Al final se utilizan 8 bytes para el sello de tiempo de petición y 8 bytes para el sello de tiempo de la respuesta del servidor. En la siguiente figura 3.21 se observa la comparación de la captura de una respuesta del servidor de 5 datos dobles con la respuesta de la petición de 10 datos de tipo doble.

Al tener una respuesta de 11 datos del tipo doble se debería tener una longitud en bytes mayor a la respuesta de 5 datos de tipo doble, esa diferencia es de 6*30 = 180 bytes. En la prueba anterior se observó un tamaño de 326 bytes que al comparar con la nueva captura de 11 datos tipo doble, se puede observar que el tamaño de la trama es de 506 bytes que es la diferencia de 6 datos de tipo doble que se tiene y la cual se puede observar en la figura 3.22 de la captura de la respuesta de la petición de 11 datos tipo doble.

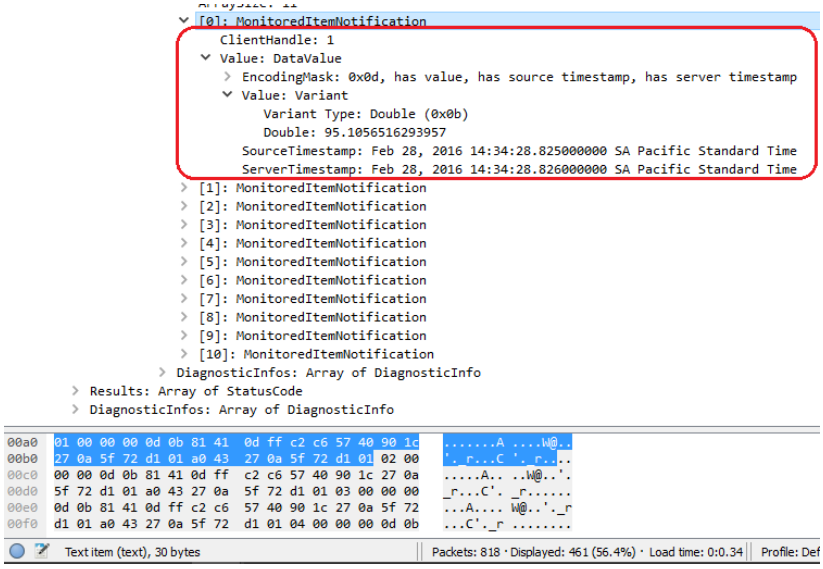

Figura 3.21: Composición de bytes utilizado en el mensaje del cuerpo del mensaje OPC UA de respuesta de un dato tipo doble. Fuente autor.

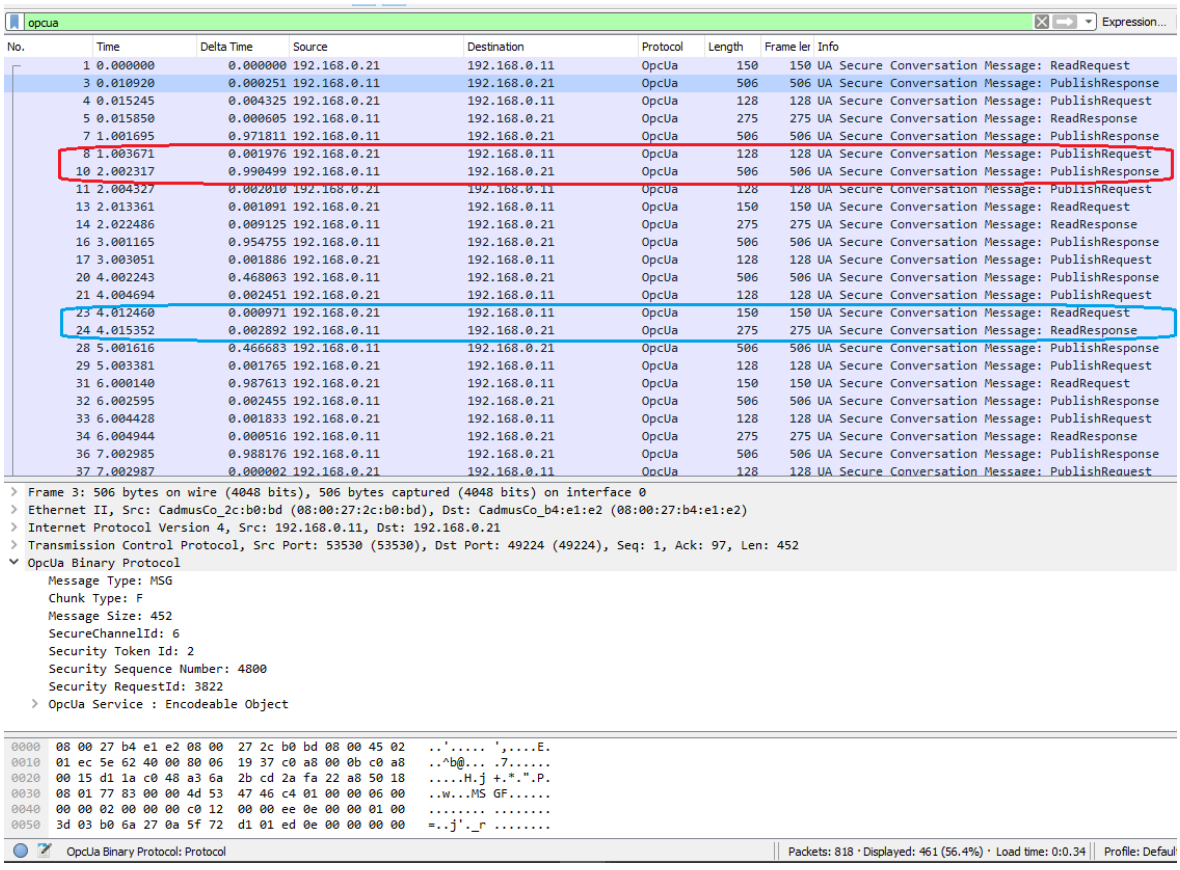

Figura 3.22: Captura del tráfico OPC UA. Escenario de prueba 1b. Fuente autor.

Al analizar la captura realizada del tráfico de datos se puede observar una trama para la petición de variables de 128 bytes, la que no ha cambiado con respecto a la petición de 5 datos dobles, y de la respuesta a esta petición de 506 bytes la cual es 180 bytes superior debido a los 6 datos de tipo doble adicionales. Mientras la petición del estado del servidor y su respuesta permanecen a 150 bytes y 275 bytes respectivamente. En la siguiente figura 3.22 se puede observar la gráfica del ancho de banda utilizado por la conversación de suscripción a variables y la del estado del servidor.

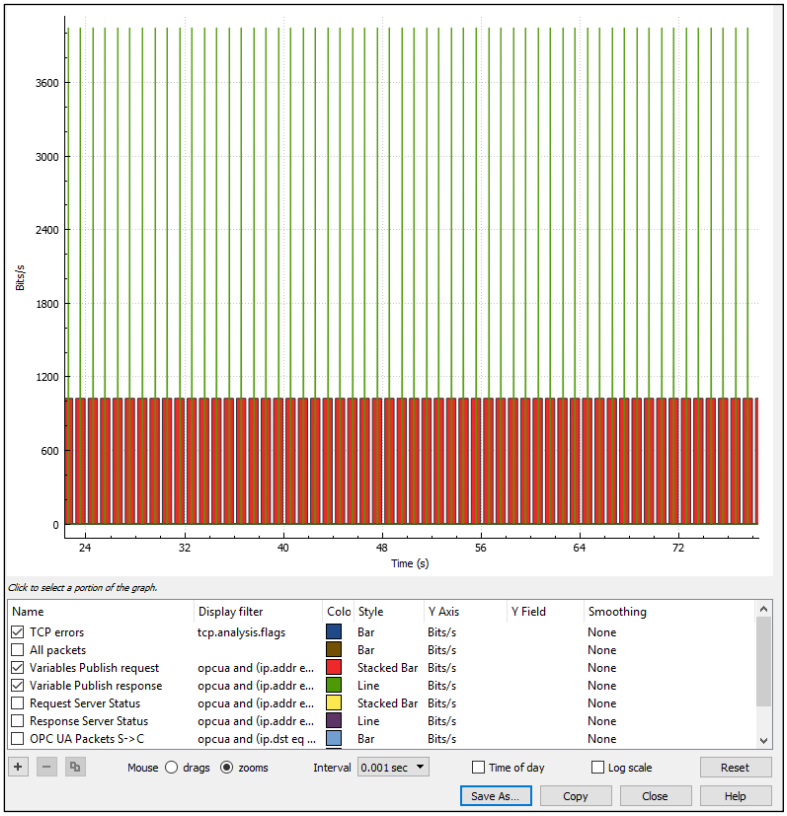

Figura 3.23: Captura del tráfico OPC UA. Escenario de prueba 1b. Fuente autor.

En el grafico anterior se puede observar un ancho de banda utilizado de 506 x 8 = 4048 bits/s para la respuesta de la lectura de las variables y de la petición permanece igual a 1024 bits/s, mientras el ancho de banda de las peticiones y del estado del servidor permanecen iguales con el ancho de banda utilizado de la respuesta de 2200 bits/s y de la petición a 1200 bits/s. Si se llegan a realizar las conversaciones simultáneas se tendría un ancho de banda utilizado de hasta 8472 bits/s.

En la figura 3.24 se observa el ancho de banda total utilizado por la conversación de las lecturas de los 11 datos dobles y de la conversación del estado del servidor.

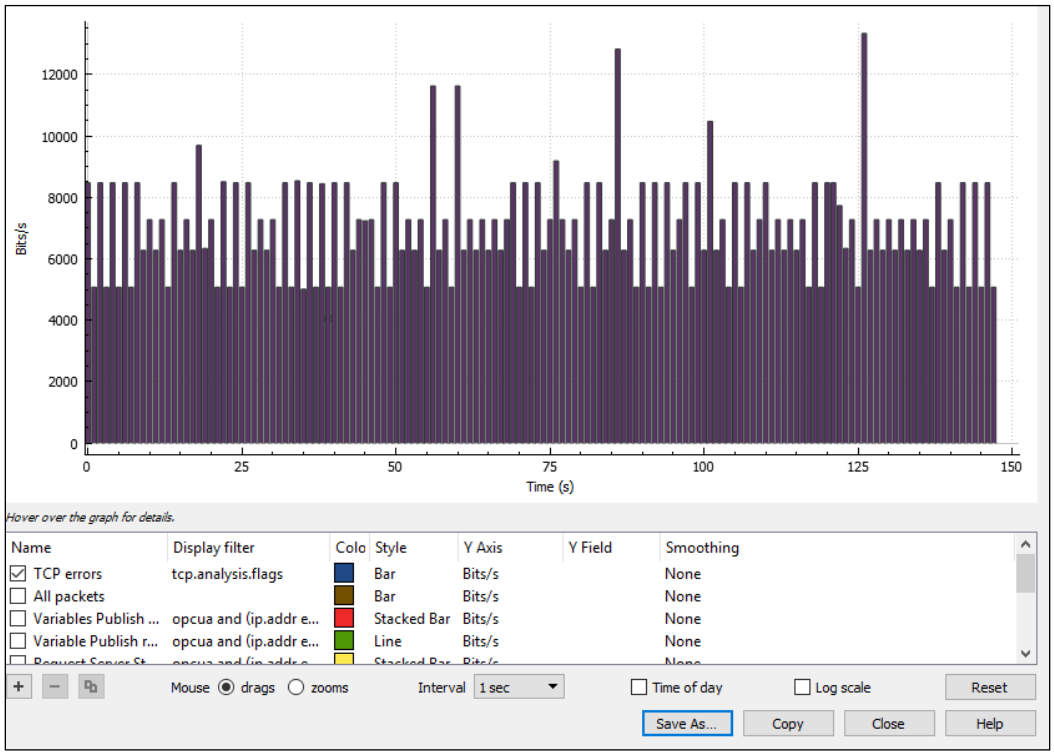

Figura 3.24: Gráfico del uso del ancho de banda del tráfico OPC UA. Escenario de prueba 1b. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	818	461 (56.4%)	N/A
Time span, s	147.012	147.004	N/A
Average pps	5.6	3.1	N/A
Average packet size, B	201.5	281.5	N/A
Bytes	164503	129606 (78.8%)	0
Average bytes/s	1118	881	N/A
Average bits/s	8951	7053	N/A

Figura 3.25: Estadísticas de la conversación OPC UA. Escenario de prueba 1b. Fuente autor.

Del ejemplo anterior se puede observar que la respuesta a la lectura de las variables OPC UA tiene una longitud de trama de 506 bytes, de los cuales se tienen para la cabecera Ethernet una longitud de 14 bytes, para la cabecera IP una longitud de 20 bytes, para la cabecera TCP una longitud de 20 bytes y para el cuerpo de la respuesta OPC UA, un valor de 452 bytes. De estos 452 bytes se tienen 11 datos dobles que se requiere 30 bytes por

cada dato y se tiene una longitud de los 11 datos tipo doble de 11*30 bytes = 330 bytes, si se resta los 452 bytes de los 330 bytes se tienen 122 bytes para las cabeceras utilizadas por el protocolo OPC UA, como son la cabecera de mensaje, la cabecera de seguridad y la cabecera de secuencia.

3.7.1.3 *Escenario de prueba 1c (Lectura de 51 datos).*

Cuando los datos pedidos en el servicio de lectura de variables tienen una longitud que supera al MTU, se produce la segmentación.

En el siguiente caso se pide la lectura de 51 datos los cuales cada uno necesita de 30 bytes con una longitud resultante de 51x30 bytes = 1530 bytes solo para los datos y se debe aumentar las cabeceras del protocolo OPC UA que se suma 122 bytes dando un total de 1652 bytes. En la siguiente figura 3.26 se puede observar como el paquete se divide en 2 segmentos, uno de 1514 bytes y otro de 246 bytes.

A comparación con la lectura de 11 datos del tipo doble del escenario anterior, en la presente prueba se tienen 1200 bytes adicionales que darían 1200x8=9600 bits/s extras al escenario de prueba de 11 datos de tipo doble dando un total de 8472 + 9600 = 18072 bits/s para la prueba con 51 datos del tipo doble. En la siguiente figura 3.27 se muestra el diagrama del ancho de banda resultante.

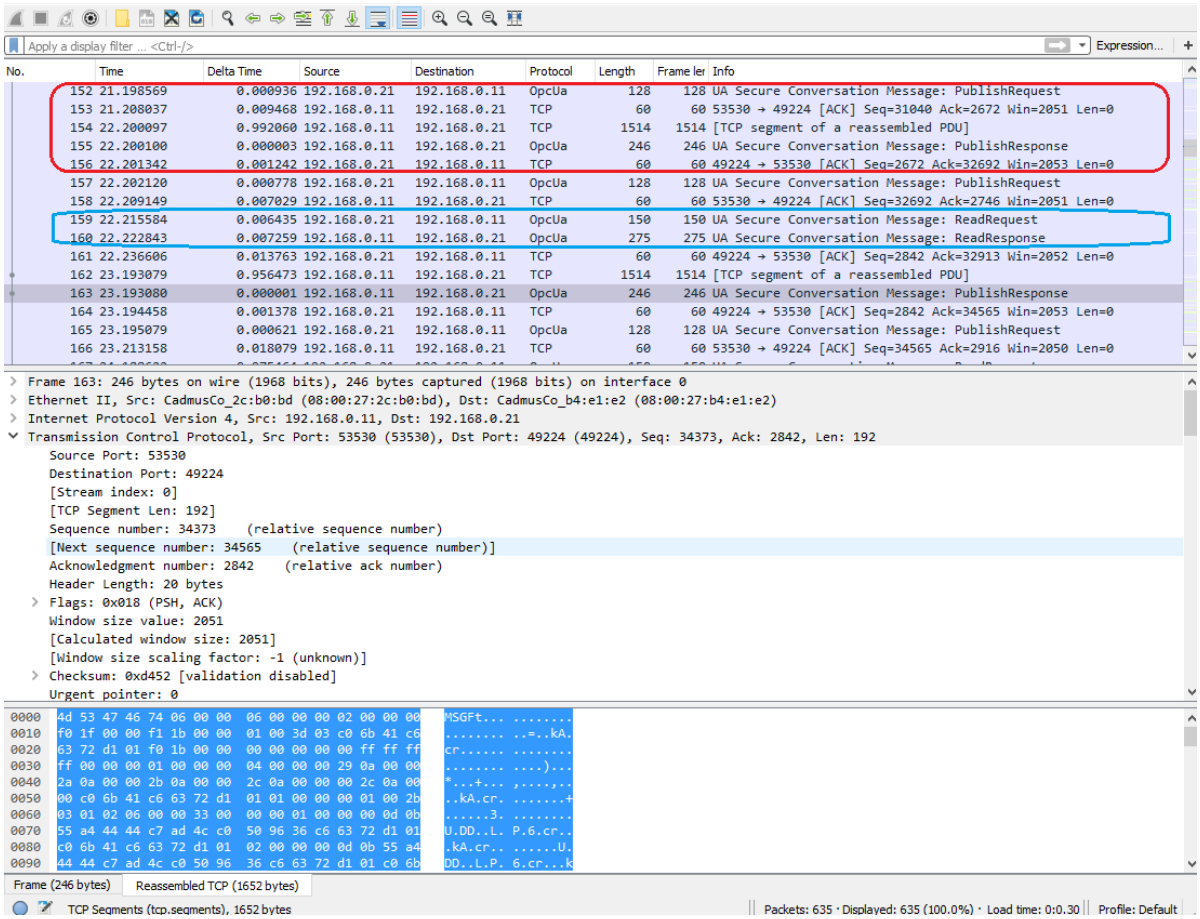

Figura 3.26: Captura del tráfico OPC UA. Escenario de prueba 1c. Fuente autor.

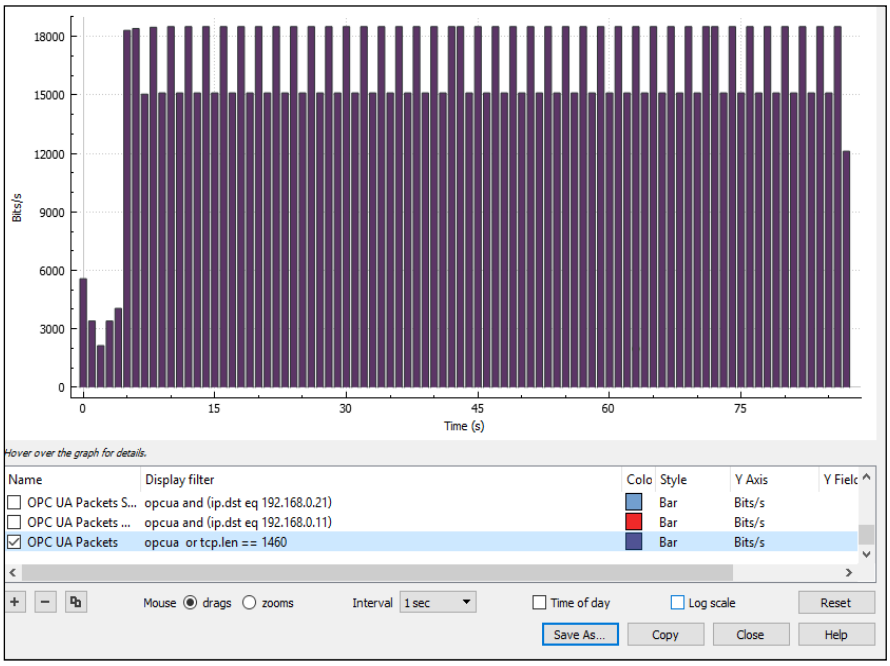

Figura 3.27: Uso de ancho de banda del tráfico OPC UA. Escenario de prueba 1c. Fuente autor.

En la figura 3.28 se muestra un resumen de las estadísticas de la captura del tráfico de red para el escenario de prueba 1 con la lectura de 51 datos del tipo doble.

Statistics

Measurement	Captured	Displayed	Marked
Packets	635	347 (54.6%)	N/A
Time span, s	87.194	87.004	N/A
Average pps	7.3	4.0	N/A
Average packet size, B	319.5	509.5	N/A
Bytes	202799	176876 (87.2%)	0
Average bytes/s	2325	2032	N/A
Average bits/s	18 k	16 k	N/A

Figura 3.28: Estadísticas de la conversación OPC UA. Escenario de prueba 1c. Fuente autor.

3.7.1.4 *Escenario de prueba 1d (Lectura de 1000 datos).*

En el siguiente escenario de prueba se realiza la lectura de 1000 datos de tipo doble. Con cada dato de tipo doble se necesita 30 bytes con lo cual se tendría una longitud total de 30000 bytes de datos por segundo y 30000 x 8 = 240000 bits/s solo para datos, adicionalmente se tienen 122 bytes de las cabeceras del protocolo OPC UA y 54 bytes de las cabeceras TCP, IP y Ethernet que al sumar se tienen 176 bytes x 8 = 1408 bits/s. También se tiene el ancho de banda utilizado para el chequeo del estado del servidor que suman 150 + 275 bytes = 425 bytes por segundo y 425 x 8 = 3400 bits/s. Si se suman todos los anchos de banda se tienen 244808 bits/s. Como se puede apreciar el mayor ancho de banda es utilizado en la lectura de los 1000 datos tipo doble.

En la siguiente figura 3.29 se puede apreciar la captura del tráfico de la lectura de los 1000 datos tipo doble, y se observa que se fragmenta el paquete en 21 segmentos los cuales en el cliente OPC UA son re-ensamblados para obtener todos los datos.

En la figura 3.30 se observa el ancho de banda utilizado para la lectura de 1000 datos de tipo doble con 1 segundo de muestreo.

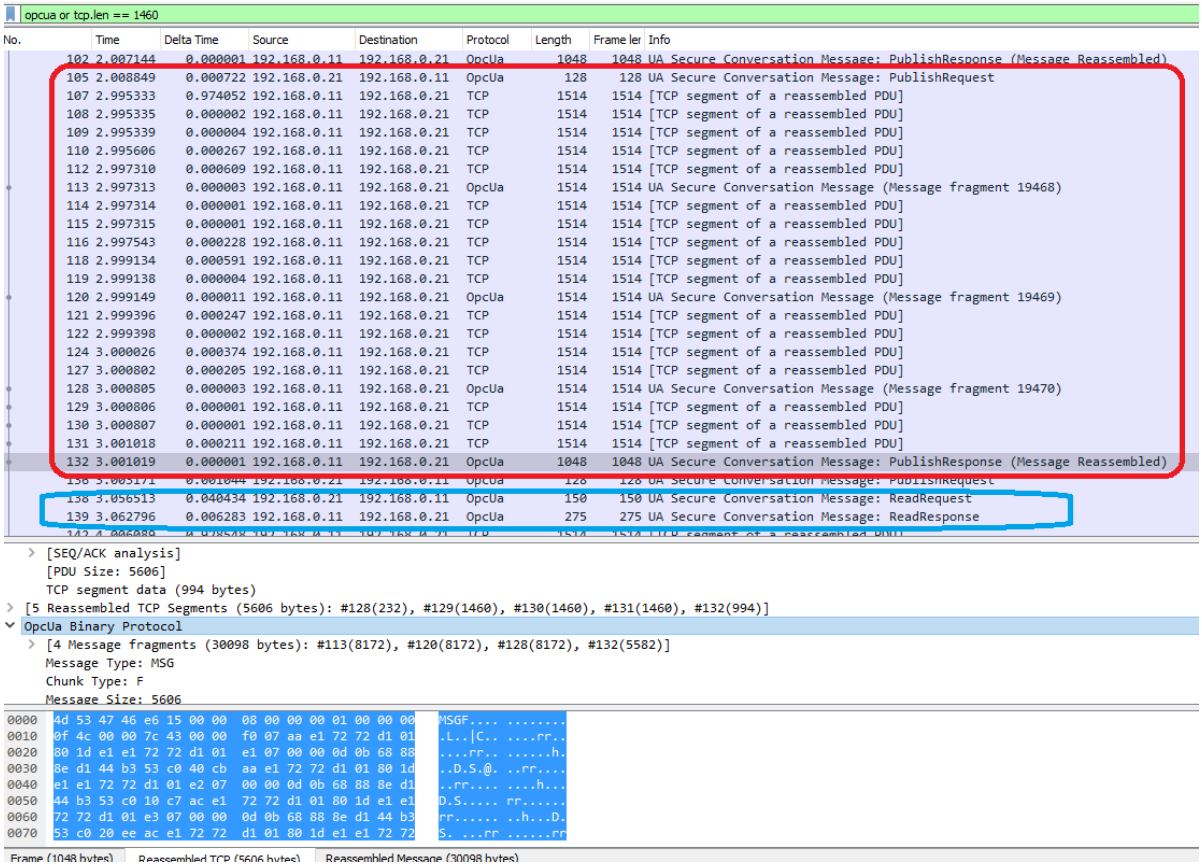

Figura 3.29: Captura el tráfico OPC UA. Escenario de prueba 1d. Fuente autor.

Figura 3.30: Uso de ancho de banda del tráfico OPC UA. Escenario de prueba 1d. Fuente autor.

En la figura 3.31 se muestra un resumen de las estadísticas de la captura del tráfico de red para el escenario de prueba 1 con la lectura de 1000 datos del tipo doble.

Statistics

Measurement	Captured	Displayed	Marked
Packets	2405	2405 (100.0%)	N/A
Time span, s	70.686	70.686	N/A
Average pps	34.0	34.0	N/A
Average packet size, B	957.5	957.5	N/A
Bytes	2303860	2303860 (100.0%)	0
Average bytes/s	32 k	32 k	N/A
Average bits/s	260 k	260 k	N/A

Figura 3.31: Estadísticas de la conversación OPC UA. Escenario de prueba 1d. Fuente autor.

3.7.2 Escenario de prueba 2.

3.7.2.1 *Escenario de prueba 2a (Lectura de 1000 datos).*

En el siguiente escenario de prueba **(Sin modo o política de seguridad, usuario anónimo y tiempo de muestreo de 500 milisegundos)**, se realiza la lectura de 1000 datos de tipo doble con un tiempo de muestreo de 500 ms, para observar la diferencia de uso del ancho de banda con el escenario de prueba de lectura de 1000 ms a 1 segundo de muestreo. Para los 1000 datos se necesita un total de 30000 bytes de datos por segundo y 30000 x 8 = 240000 bits y debido a que se lee dos veces por segundo se tendría 240000 x 2 = 480000 bits/s solo para datos, adicionalmente se tienen 122 bytes de las cabeceras del protocolo OPC UA y 54 bytes de las cabeceras TCP, IP y Ethernet que al sumar se tienen 176 bytes x 8 = 1408 bits/s. También se tiene el ancho de banda utilizado para el chequeo del estado del servidor que suman 150 + 275 bytes = 425 bytes por segundo y 425 x 8 = 3400 bits/s, estos datos permanecen igual aunque se cambie el periodo de muestreo, debido a que esta característica la realiza durante cada segundo independiente del tiempo de muestreo de lectura de los datos. Si se suman todos los anchos de banda se tienen 484808 bits/s. Como se puede apreciar el mayor ancho de banda es utilizado en la lectura de los 1000 datos tipo doble.

Como se puede apreciar en la captura de tráfico de datos, se observa que en un segundo hay dos servicios de petición de publicación y de respuesta de publicación en la lectura de los datos.

En la siguiente figura 3.32 se puede apreciar la captura del tráfico de la lectura de los 1000 datos tipo doble, y se observa que se fragmenta el paquete en 21 segmentos los cuales en el cliente OPC UA son re-ensamblados para obtener todos los datos.

En la figura 3.33 se observa el ancho de banda utilizado para la lectura de 1000 datos de tipo doble con 500 milisegundos de muestreo.

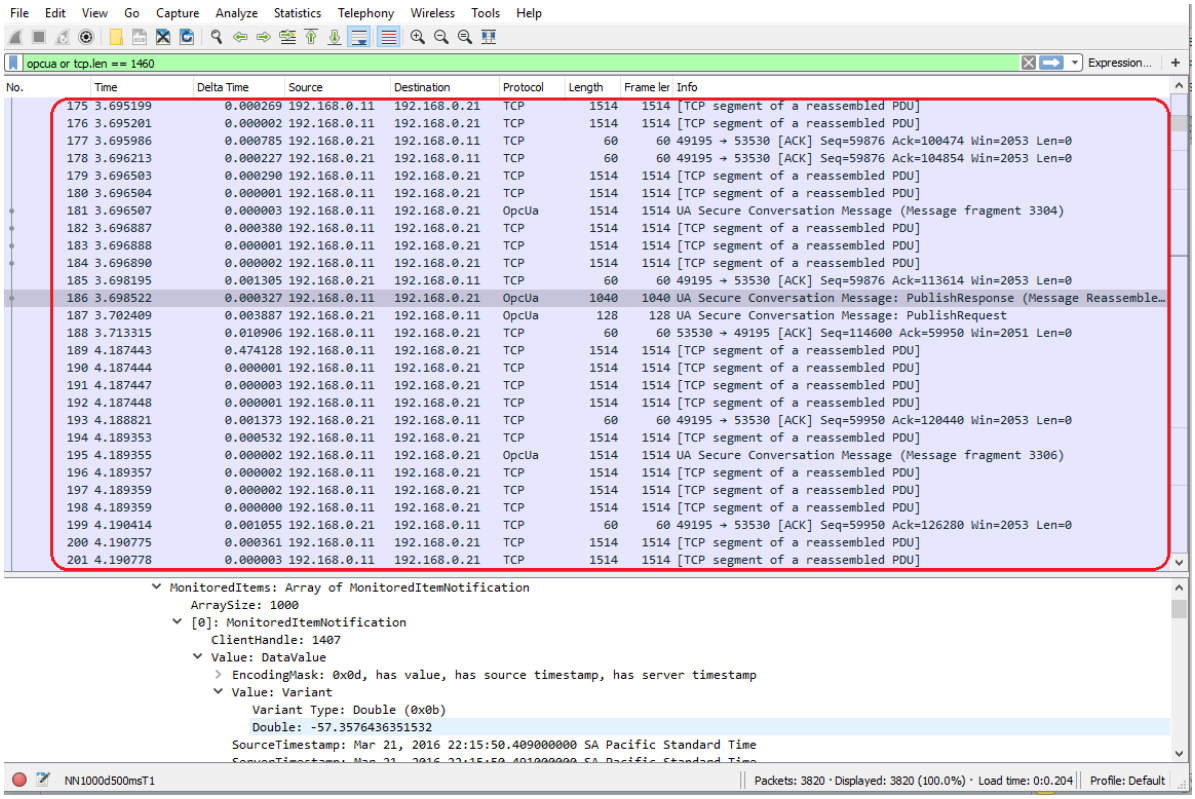

Figura 3.32: Captura el tráfico OPC UA. Escenario de prueba 2a. Fuente autor.

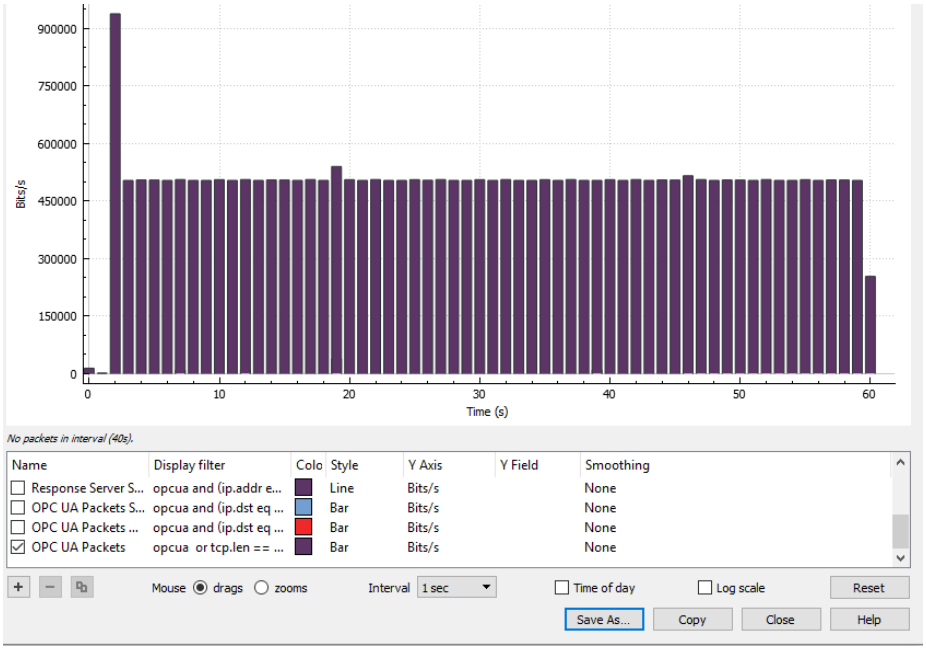

Figura 3.33: Uso de ancho de banda del tráfico OPC UA. Escenario de prueba 2a. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	3820	3820 (100.0%)	N/A
Time span, s	60.189	60.189	N/A
Average pps	63.5	63.5	N/A
Average packet size, B	1000.5	1000.5	N/A
Bytes	3820538	3820538 (100.0%)	0
Average bytes/s	63 k	63 k	N/A
Average bits/s	507 k	507 k	N/A

Figura 3.34: Estadísticas de la conversación OPC UA. Escenario de prueba 2a. Fuente autor.

3.7.3 Escenario de prueba 3.

3.7.3.1 *Escenario de prueba 3a (Lectura de 1000 datos).*

En el siguiente escenario de prueba **(Sin modo o política de seguridad, usuario anónimo y tiempo de muestreo de 100 milisegundos),** se realiza la lectura de 1000 datos de tipo doble con un tiempo de muestreo de 100 ms. Debido a que se lee diez veces por segundo se tendría 240000 x 10 = 2.4 Mbps solo para datos. Si se suman todos los anchos de banda se tienen 2.484808 Mbps. Como se puede apreciar el mayor ancho de banda es utilizado en la lectura de los 1000 datos tipo doble a 100 ms.

En el gráfico de la captura del tráfico de datos, se observa que en un segundo hay 1 servicio de petición de publicación y 1 respuesta de publicación en cada 100 milisegundos para la lectura de los datos.

En la siguiente figura 3.34 se puede apreciar la captura del tráfico de la lectura de los 1000 datos tipo doble, y se observa que se fragmenta el paquete en 21 segmentos los cuales en el cliente OPC UA son re-ensamblados para obtener todos los datos.

En la figura 3.35 se observa el ancho de banda utilizado para la lectura de 1000 datos de tipo doble con 100 milisegundos de muestreo.

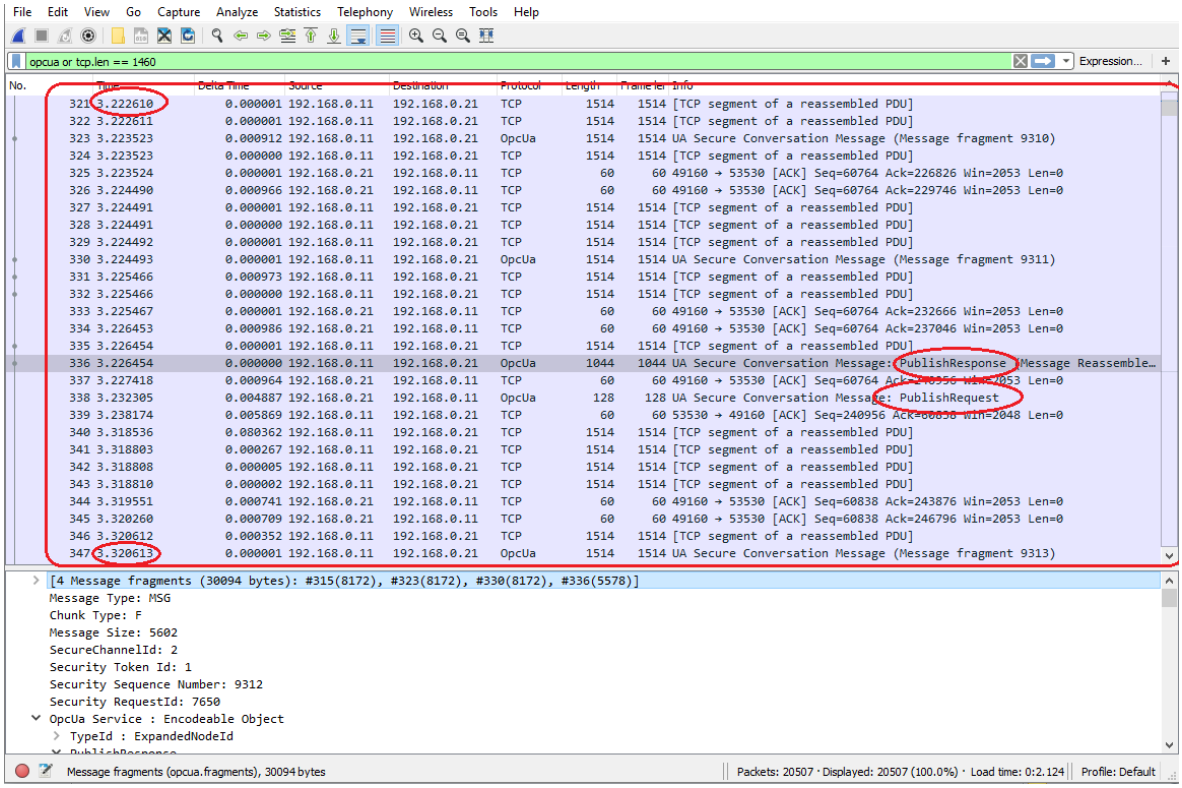

Figura 3.35: Captura el tráfico OPC UA. Escenario de prueba 3a. Fuente autor.

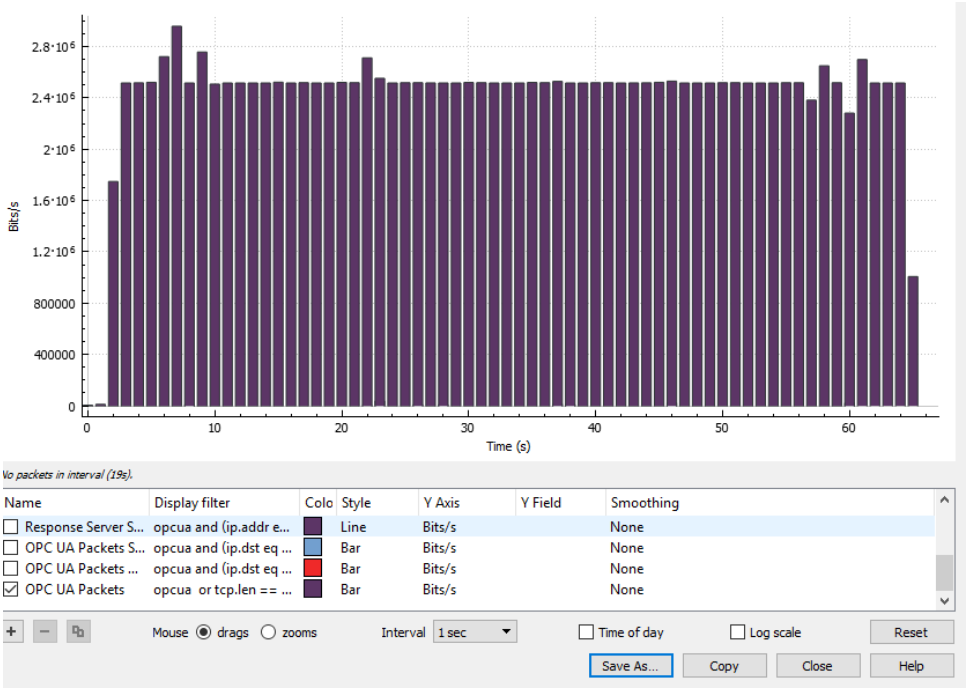

Figura 3.36: Uso de ancho de banda del tráfico OPC UA. Escenario de prueba 3a. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	20507	20507 (100.0%)	N/A
Time span, s	65.328	65.328	N/A
Average pps	313.9	313.9	N/A
Average packet size, B	993.5	993.5	N/A
Bytes	20363900	20363900 (100.0%)	0
Average bytes/s	311 k	311 k	N/A
Average bits/s	2493 k	2493 k	N/A

Figura 3.37: Estadísticas de la conversación OPC UA. Escenario de prueba 3a. Fuente autor.

3.7.4 Escenario de prueba 4.

3.7.4.1 *Escenario de prueba 4a (Lectura de 1000 datos).*

En el siguiente escenario de prueba **(Sin modo o política de seguridad, usuario con contraseña y tiempo de muestreo de 100 milisegundos)**, en donde se realiza la activación de la sesión y la autorización de acceso mediante el uso de un usuario y contraseña se observa que el tamaño de los datos en la petición de lectura no tiene un cambio debido a que el manejo de usuario y contraseña solo se da en el servicio de

activación de la sesión y verificación de la autorización de usuarios en el servidor OPC. Se puede observar que para la petición de los 1000 datos dobles a 100 milisegundos se utiliza el mismo ancho de banda de 2.4 Mbps como se puede observar en la figura 3.40.

Para configurar la autorización de un usuario en un servidor se debe crear la cuenta del usuario y su contraseña, y desde el cliente activar la sesión con el usuario y contraseña registrado en el servidor OPC UA. En la figura 3.38 se muestra la cuenta del usuario creada en el servidor OPC UA y en la figura 3.39 la configuración en el cliente OPC UA para la activación de la sesión mediante usuario y contraseña.

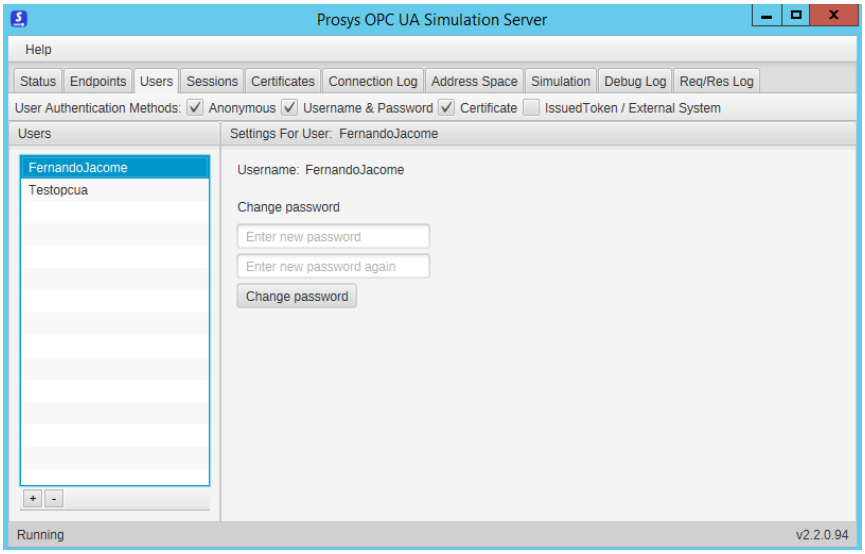

Figura 3.38: Configuración de una cuenta de usuario con contraseña en el servidor OPC UA.

Fuente autor.

En la figura 3.41 se muestra la captura de los datos para la activación de la sesión y dentro del cuerpo del mensaje OPC UA se puede observar el campo en donde se encuentra el usuario con la contraseña la cual está cifrada.

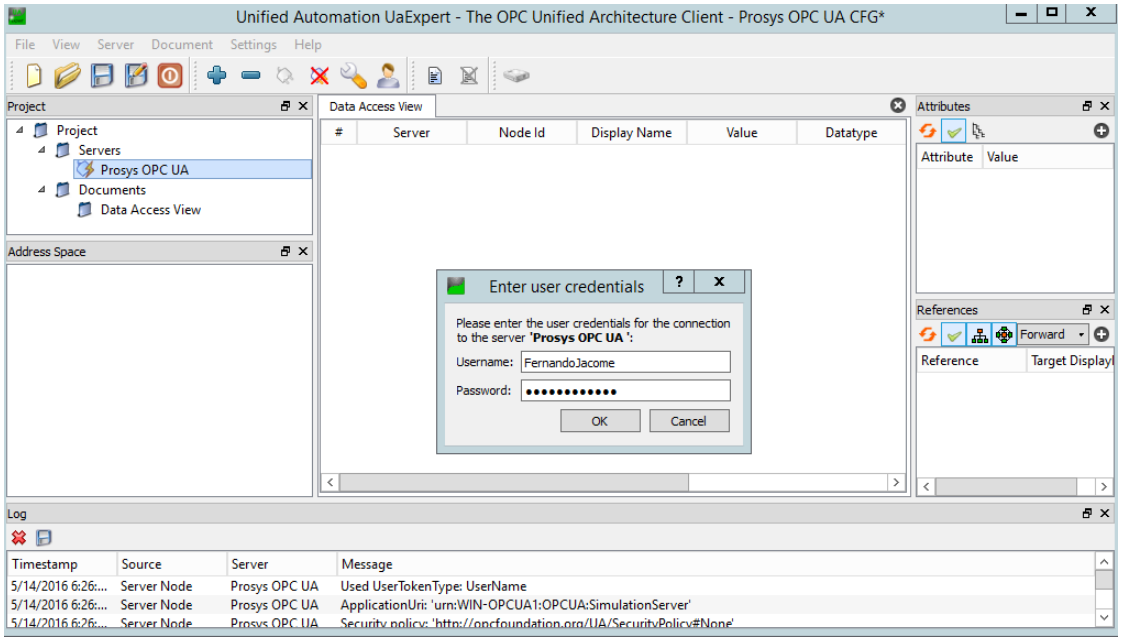

Figura 3.39: Configuración de una cuenta de usuario con contraseña en cliente OPC UA.

Fuente autor.

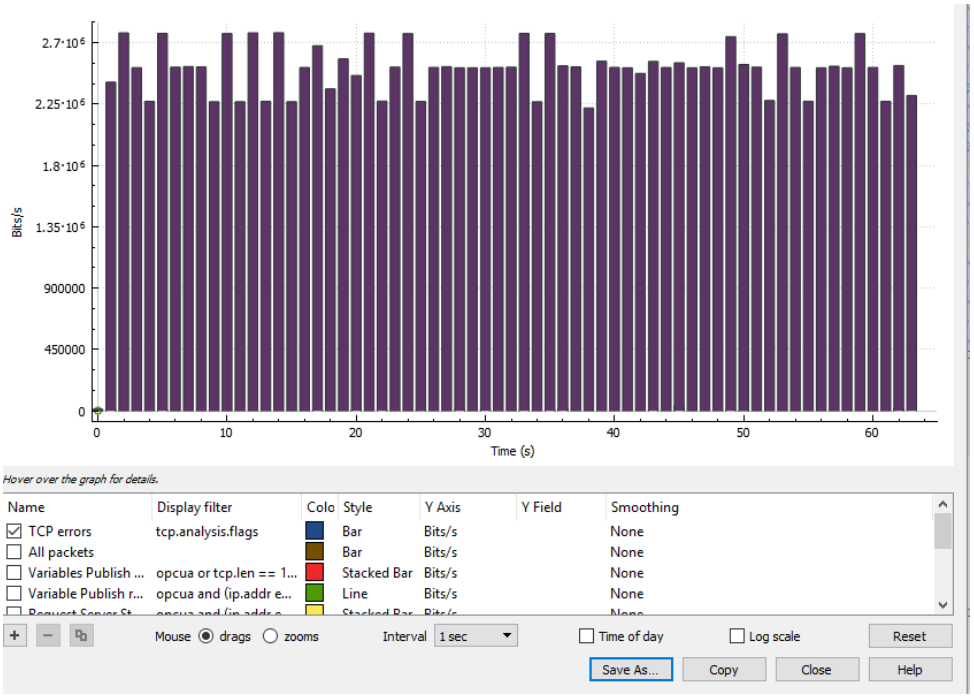

Figura 3.40: Uso de ancho de banda del tráfico OPC UA. Escenario de prueba 4a. Fuente autor.

En la figura 3.41 de la captura del tráfico de datos para la activación de la sesión OPC UA se observa que usa 499 bytes como petición de la activación y 150 bytes como respuesta a esta petición, esta cantidad de bytes es inferior a los 30000 bytes que se usa

para los datos en la petición de lectura, por lo tanto siguen predominando como uso del ancho de banda.

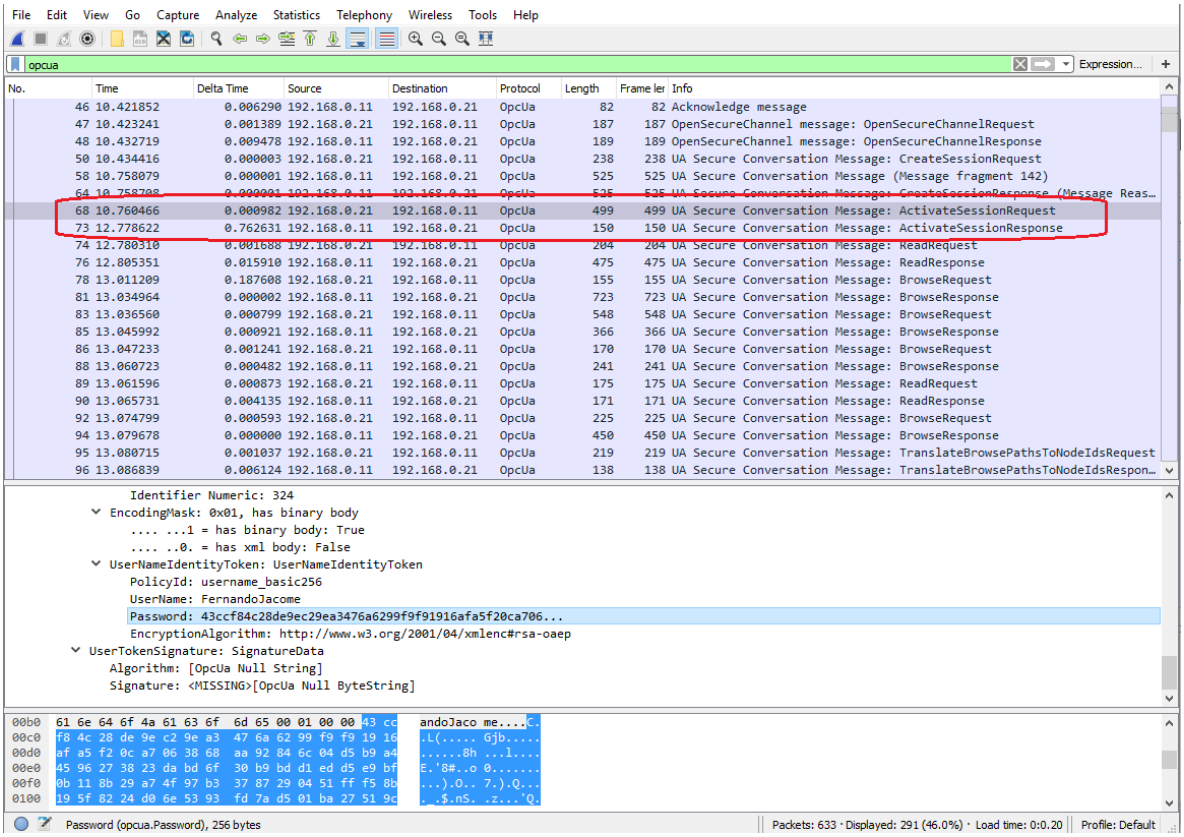

Figura 3.41: Captura el tráfico OPC UA. Escenario de prueba 4a. Fuente autor.

3.7.5 Escenario de prueba 5.

3.7.5.1 *Escenario de prueba 5a (Lectura de 1000 datos).*

En este escenario de prueba **(Sin modo de seguridad, usuario con certificado y tiempo de muestreo de 100 milisegundos),** de igual forma que el escenario anterior, el ancho de banda no varía en la lectura de los 1000 datos tipo doble y únicamente varia en el tamaño de la trama de la petición y la activación de la sesión OPC UA que es de 1763 bytes debido a la longitud del certificado y que se puede apreciar en la figura 3.42, en donde se puede observar el método de petición de activación de la sesión por medio de un certificado X509v02.

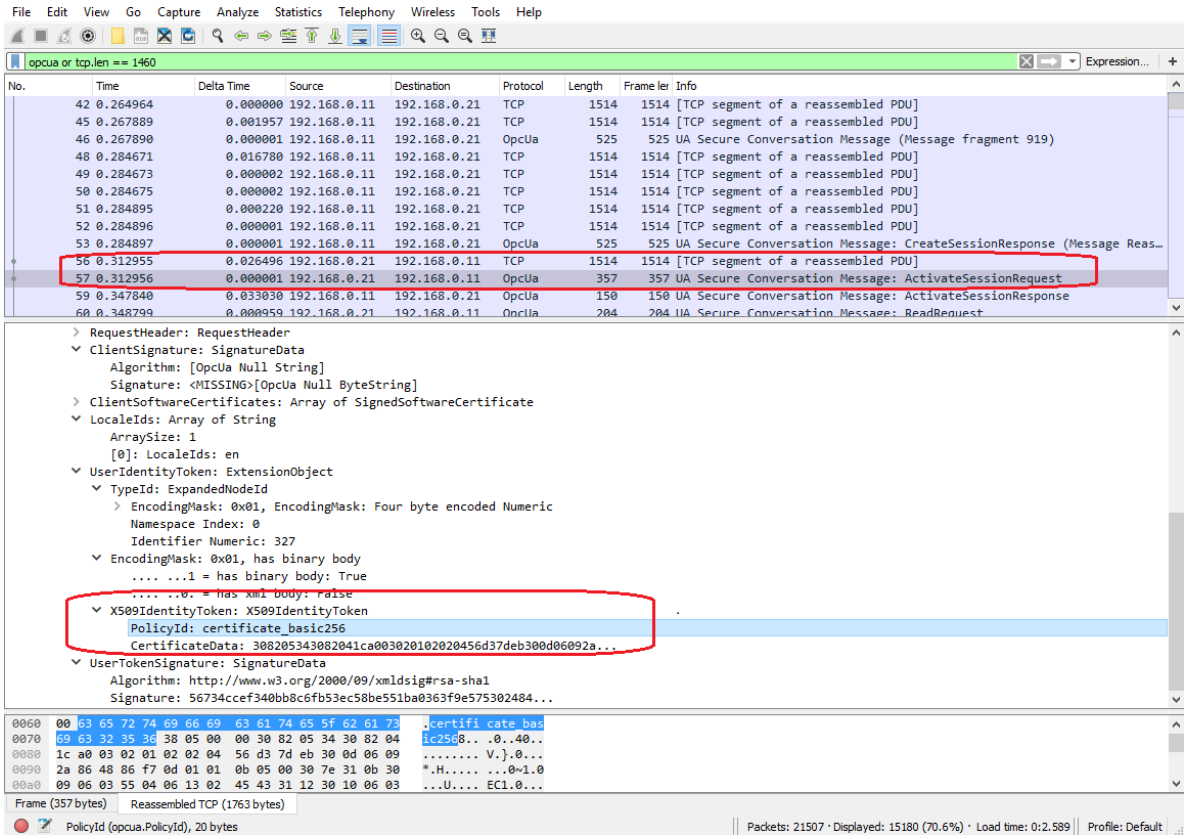

Figura 3.42: Captura el tráfico OPC UA. Escenario de prueba 5a. Petición de la activación de la sesión con autorización por certificado X509. Fuente autor.

En la figura 4.43 se puede observar el ancho de banda utilizado por la lectura de los 1000 datos a 100 milisegundos que es de 2.4 Mbps y no difiere con respecto al escenario anterior.

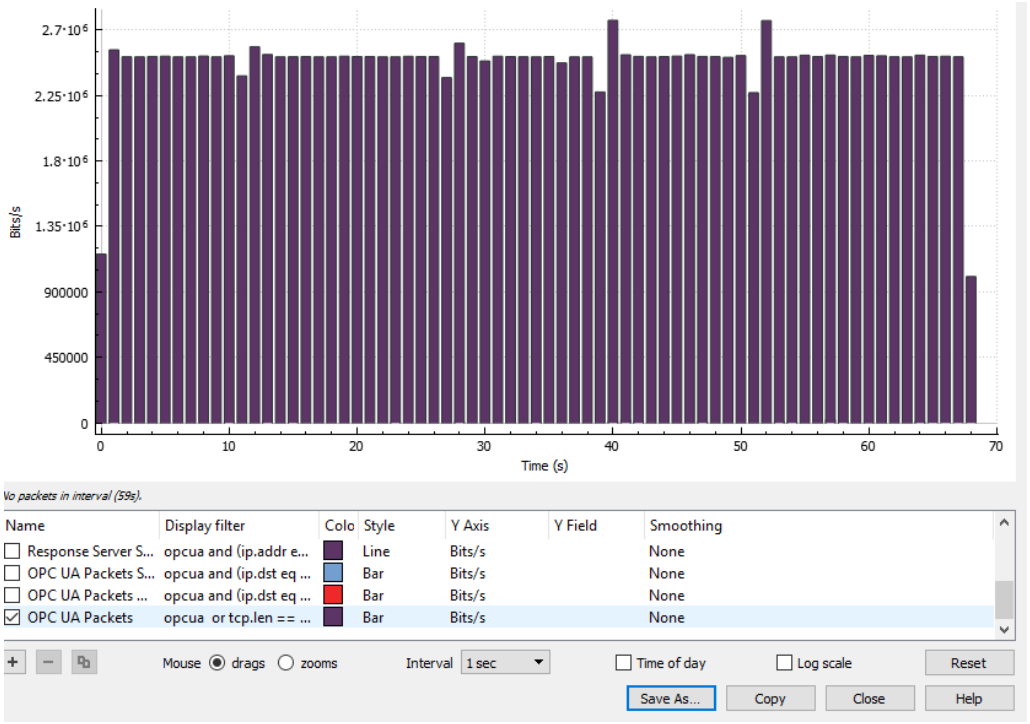

Figura 3.43: Gráfico del uso del ancho de banda del tráfico OPC UA. Escenario de prueba 5a.

Fuente autor.

3.7.6 Escenario de prueba 6.

3.7.6.1 *Escenario de prueba 6a (Lectura de 6 datos).*

En este escenario de prueba se realiza la conexión con **mensaje firmado y política de seguridad Basic128RSA15, usuario anónimo y tiempo de muestreo de 1 segundo**. Al momento de realizar la comunicación OPC UA mediante un firmado digital, se añaden 20 bytes al final del mensaje OPC UA para el manejo de este firmado digital, el cual se puede observar en la figura 3.44.

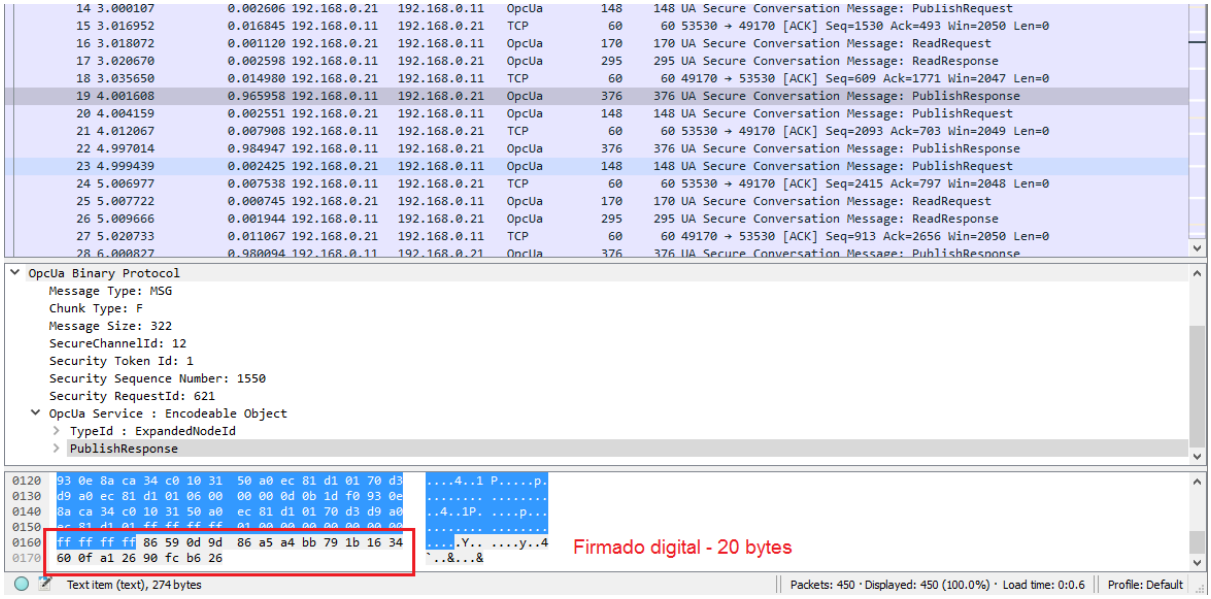

Figura 3.44: Uso de 20 bytes al final del mansaje para el firmado digital. Escenario de prueba 6a. Fuente autor.

Con respecto al ancho de banda como solo se tiene un segmento de datos al momento de la respuesta de la lectura de 6 datos de tipo doble durante 1 segundo, se tiene un ancho de banda de 6 x 30 bytes = 180 bytes y al adicionar los 122 bytes de las cabeceras OPC UA, 20 bytes del firmado digital y 54 bytes de las cabeceras TCP, IP y Ethernet, se tiene un resultado de 376 bytes durante 1 segundo o 376 x 8 = 3008 bps. Adicionalmente, se tiene el ancho de banda utilizado para ver el estado del servidor OPC UA que es de 275 bytes más 20 bytes del firmado digital dando un resultado de 295 bytes o 295 x 8 = 2360 bps, dando un total de 5368 bps. De igual forma, los 20 bytes del firmado digital se adicionan a la petición del estado del servidor con 170 bytes y a la petición de la lectura de los 6 datos de tipo doble con 148 bytes dando un total de 318 bytes o 310 x 8 = 2544 bps. En la figura 3.45 se puede observar el tráfico de datos capturado en este escenario de prueba.

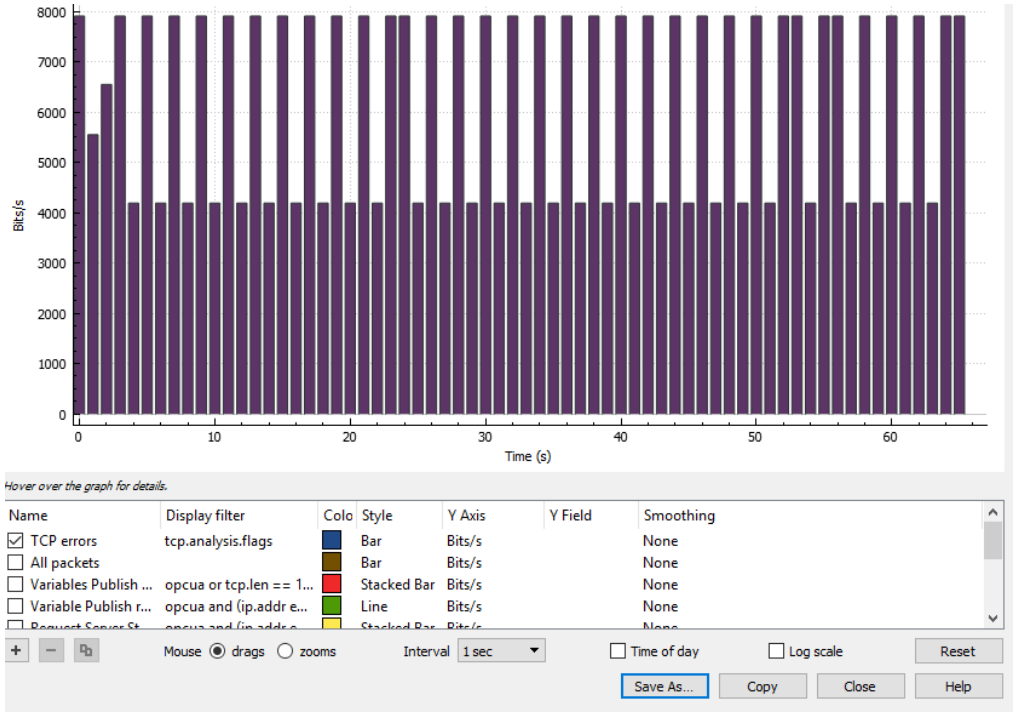

Figura 3.45: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 6a. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	382	204 (53.4%)	N/A
Time span, s	65.076	65.062	N/A
Average pps	5.9	3.1	N/A
Average packet size, B	181.5	251.5	N/A
Bytes	69502	51324 (73.8%)	0
Average bytes/s	1068	788	N/A
Average bits/s	8544	6310	N/A

Figura 3.46: Estadísticas de la conversación OPC UA. Escenario de prueba 6a. Fuente autor.

3.7.6.2 *Escenario de prueba 6b (Lectura de 100 datos).*

En el siguiente escenario se tiene la lectura de 100 datos tipo doble a 1 segundo con modo de seguridad firmado, en este caso se necesitan 3000 bytes para los datos más 122 bytes para la cabecera OPC UA, 20 bytes para la firma digital, y 54 bytes para las cabeceras de TCP, IP, y Ethernet, que en total resultan 3194 bytes o 3194 x 8 = 25552 bps. Como se necesita más de 1460 bytes del MTU, el mensaje es segmentado en 3 pedazos, dos de 1460 y uno de 222, más los 54 bytes en cada mensaje de las cabeceras de TCP, IP y Ethernet. Además, se debe considerar los 465 bytes utilizados en verificar el estado del servidor y los 148 bytes para la petición de lectura de los datos. En la siguiente figura se muestra la captura de tráfico del escenario de prueba.

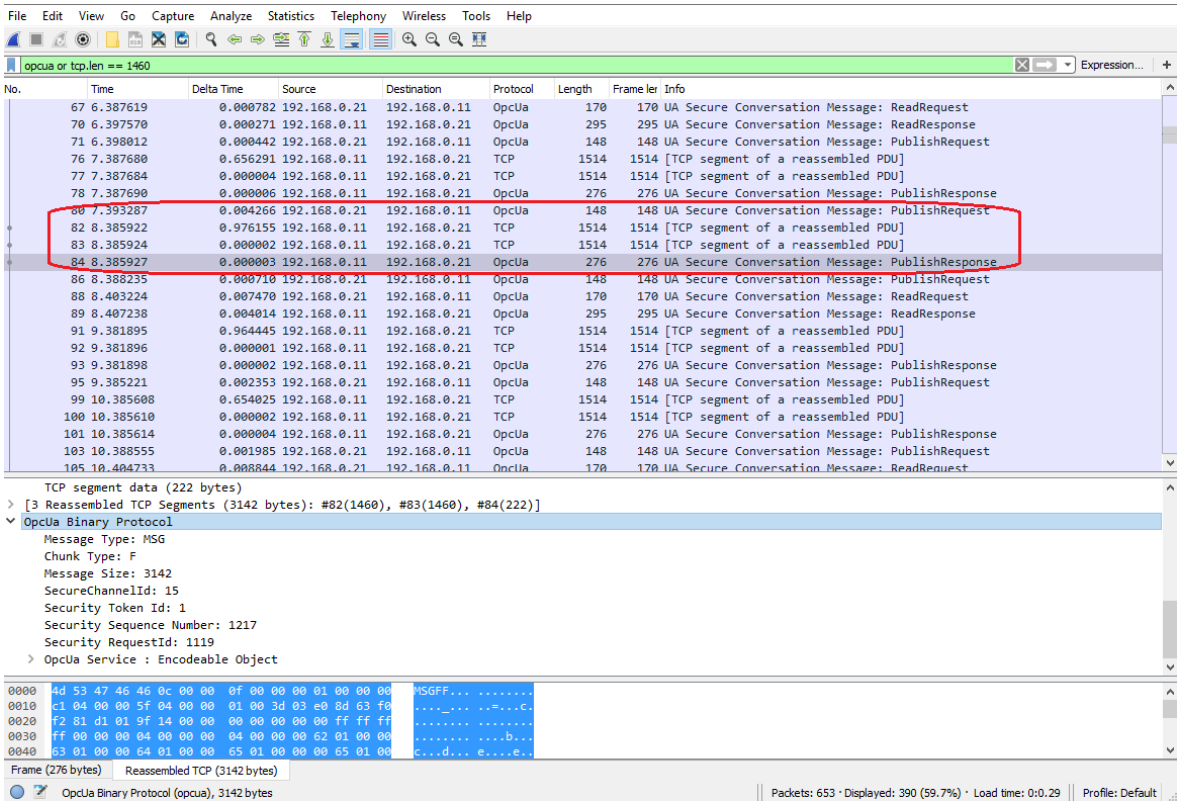

Figura 3.47: Captura el tráfico OPC UA. Escenario de prueba 6b. Fuente autor.

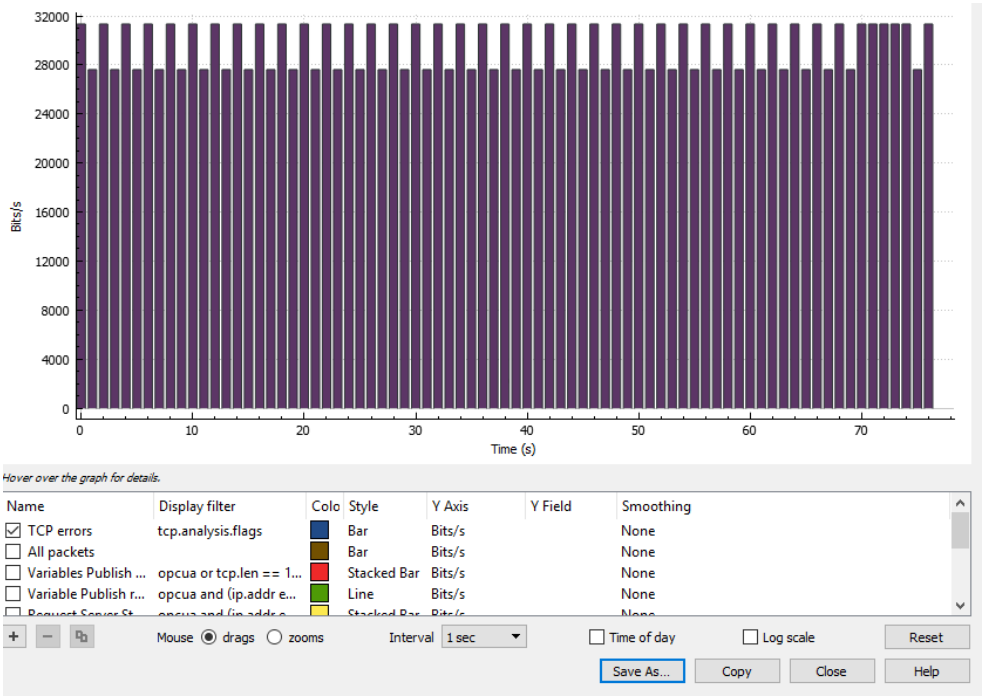

Figura 3.48: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 6b. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	653	390 (59.7%)	N/A
Time span, s	77.281	76.016	N/A
Average pps	8.4	5.1	N/A
Average packet size, B	470.5	730.5	N/A
Bytes	307485	284869 (92.6%)	0
Average bytes/s	3978	3747	N/A
Average bits/s	31 k	29 k	N/A

Figura 3.49: Estadísticas de la conversación OPC UA. Escenario de prueba 6b. Fuente autor.

3.7.7 Escenario de prueba 7.

3.7.7.1 *Escenario de prueba 7a (Lectura de 100 datos).*

En el siguiente escenario **(Con mensaje firmado y política de seguridad Basic128RSA15, usuario anónimo y tiempo de muestreo de 500 milisegundos),** se tiene la lectura de 100 datos tipo doble a 500 milisegundos con modo de seguridad firmado, en este caso se necesitan 3000 bytes para los datos más 122 bytes para la cabecera OPC UA, 20 bytes para la firma digital, y 54 bytes para las cabeceras de TCP, IP, y Ethernet, que en total resultan 3194 bytes o 3194 x 8 = 25552 bits tomando en cuenta dos mensajes por segundo se tiene 51104 bps. Como se necesita más de 1460 bytes del MTU, el mensaje es segmentado en 3 pedazos, dos de 1460 y uno de 222, más los 54 bytes en cada mensaje de las cabeceras de TCP, IP y Ethernet. En la siguiente figura se muestra la captura de tráfico del escenario de prueba.

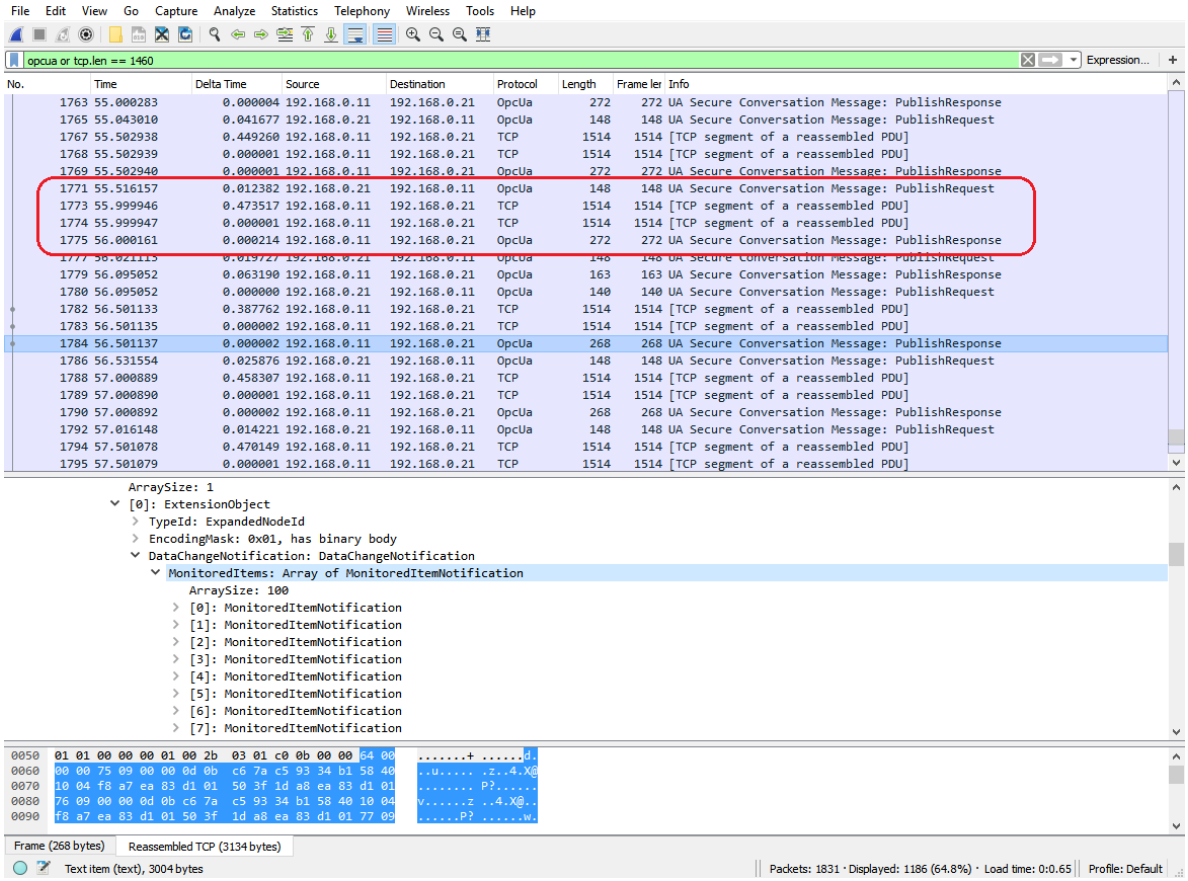

Figura 3.50: Captura el tráfico OPC UA. Escenario de prueba 7a. Fuente autor.

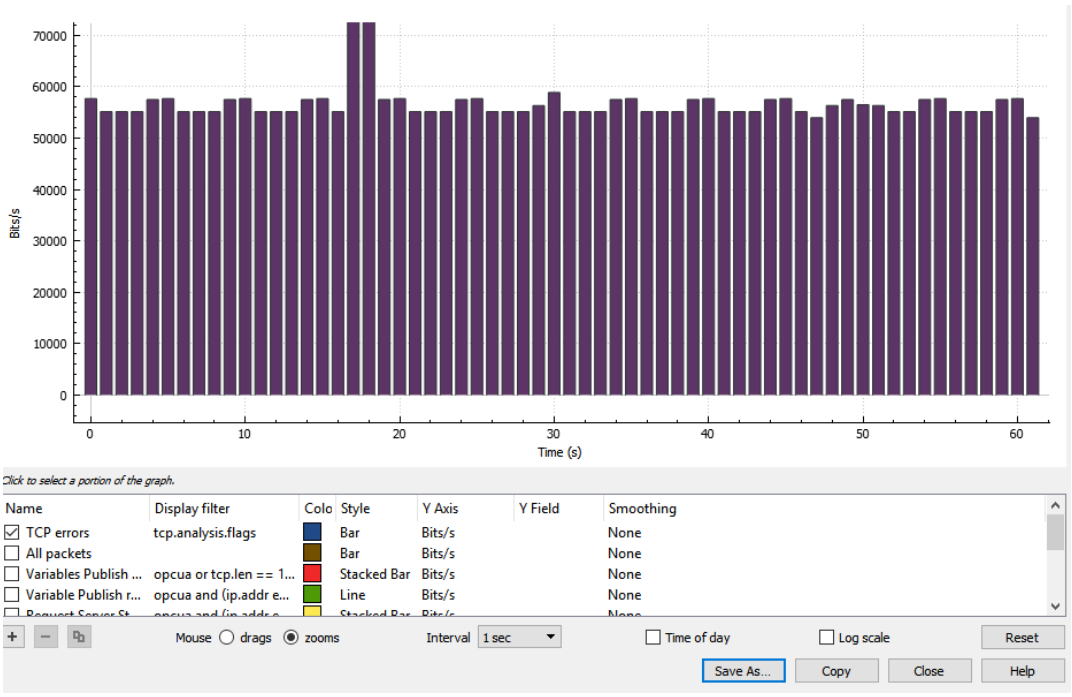

Figura 3.51: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 7a. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	963	611 (63.4%)	N/A
Time span, s	61.958	61.957	N/A
Average pps	15.5	9.9	N/A
Average packet size, B	500.5	746.5	N/A
Bytes	482302	456099 (94.6%)	0
Average bytes/s	7784	7361	N/A
Average bits/s	62 k	58 k	N/A

Figura 3.52: Estadísticas de la conversación OPC UA. Escenario de prueba 7a. Fuente autor.

3.7.8 Escenario de prueba 8.

3.7.8.1 *Escenario de prueba 8a (Lectura de 100 datos).*

En el siguiente escenario **(Con mensaje firmado y política de seguridad Basic128RSA15, usuario anónimo y tiempo de muestreo de 100 milisegundos),** se tiene la lectura de 100 datos tipo doble a 100 milisegundos con modo de seguridad firmado, en este caso se necesitan 3194 bytes o 3194 x 8 = 25552 bits tomando en cuenta 10 mensajes por segundo se tiene 255520 bps. En la siguiente figura se muestra la captura de tráfico del escenario de prueba.

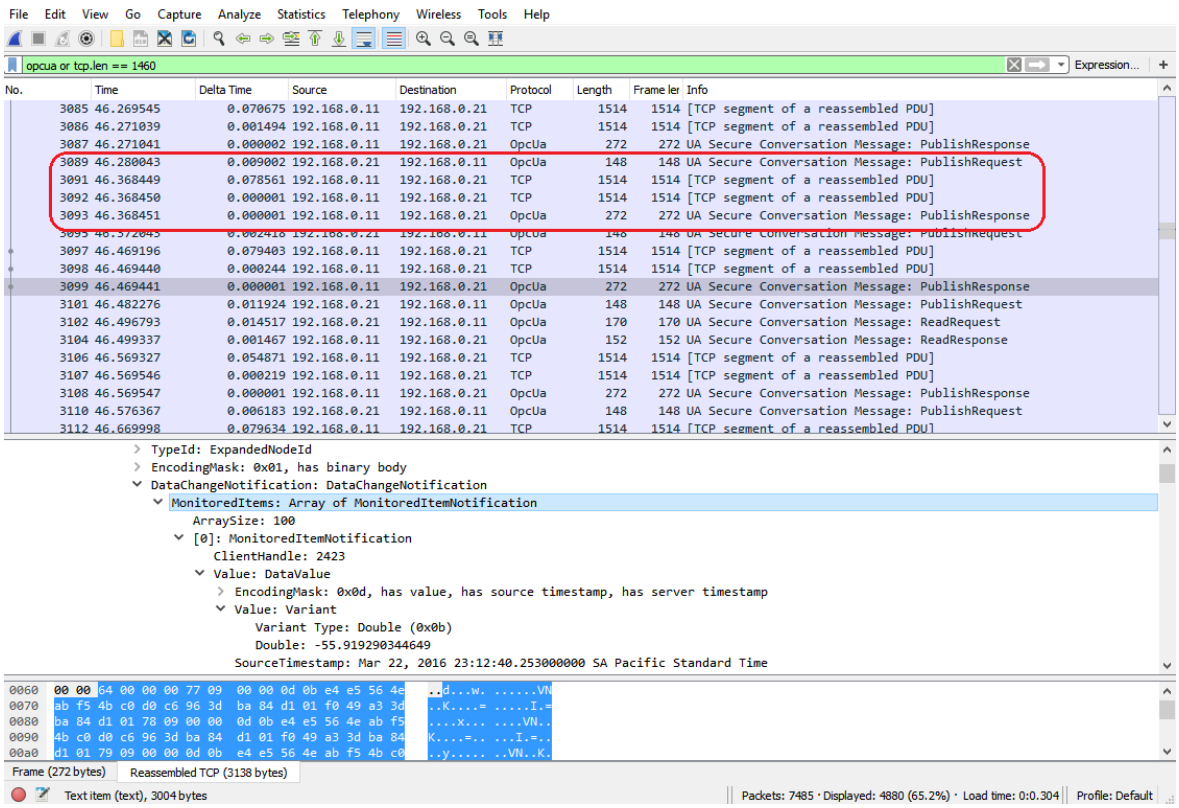

Figura 3.53: Captura el tráfico OPC UA. Escenario de prueba 8a. Fuente autor.

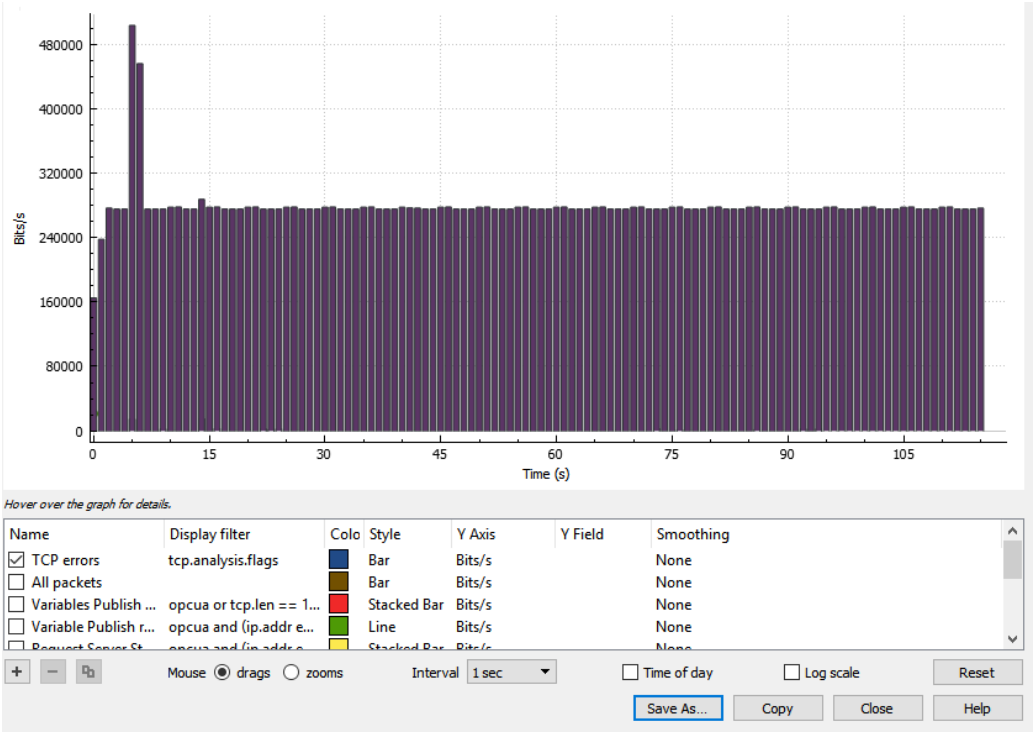

Figura 3.54: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 8a. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	7485	4880 (65.2%)	N/A
Time span, s	115.970	115.956	N/A
Average pps	64.5	42.1	N/A
Average packet size, B	562.5	829.5	N/A
Bytes	4211097	4046991 (96.1%)	0
Average bytes/s	36 k	34 k	N/A
Average bits/s	290 k	279 k	N/A

Figura 3.55: Estadísticas de la conversación OPC UA. Escenario de prueba 8a. Fuente autor.

3.7.9 Escenario de prueba 9

3.7.9.1 *Escenario de prueba 9a (Lectura de 100 datos).*

En el siguiente escenario se tiene la conexión **con mensaje firmado y política de seguridad Basic256, usuario anónimo y tiempo de muestreo de 1 segundo**. A diferencia de la política de seguridad que utiliza el algoritmo Basic128Rsa15, el algoritmo Basic256 utiliza una llave para cifrar con 256 bits y no utiliza el algoritmo RSA 1.5, en lugar de este,

utiliza RSA-OAEP. En general es más seguro un mensaje firmado con una clave de 256 bits aunque para descifrar el mensaje necesita de un poco más de procesamiento.

En la respuesta de los 100 datos de tipo doble se observa la misma cantidad de 20 bytes para la firma digital del mensaje y la cantidad de bytes transmitidos que son de 3194 bytes, como se puede observar en la figura 3.56.

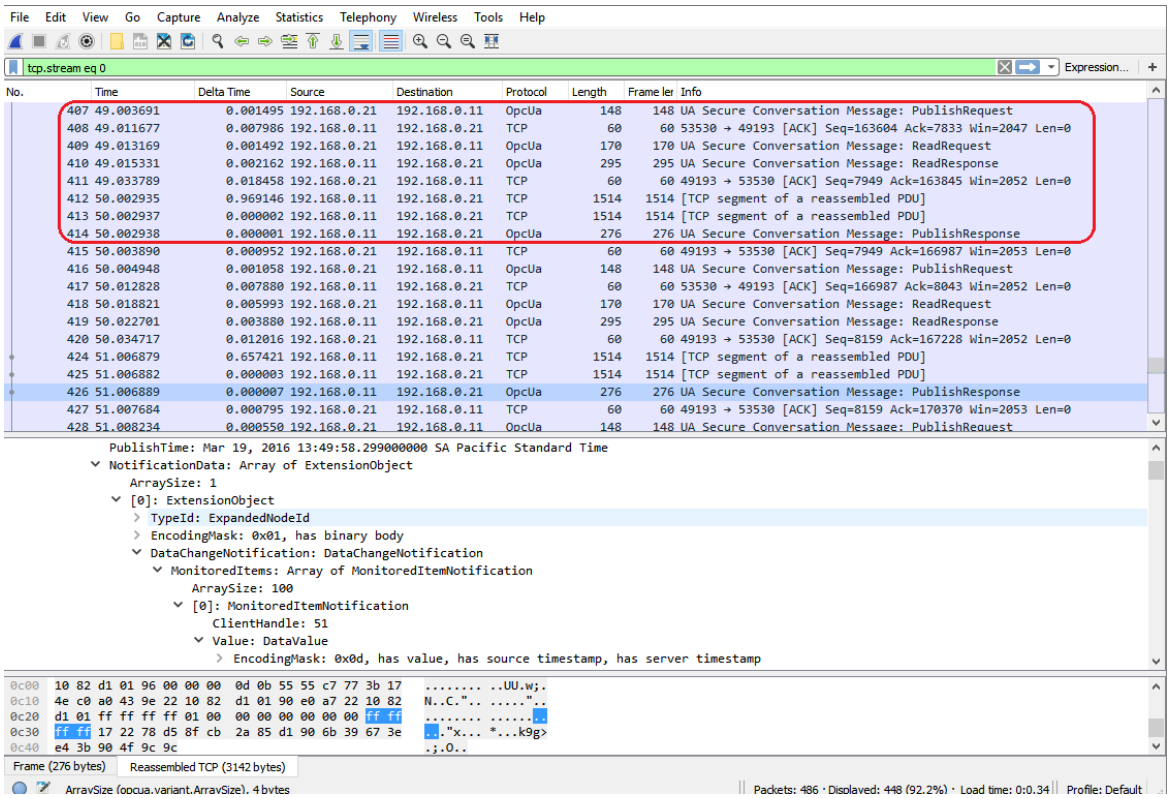

Figura 3.56: Captura el tráfico OPC UA. Escenario de prueba 9a. Fuente autor.

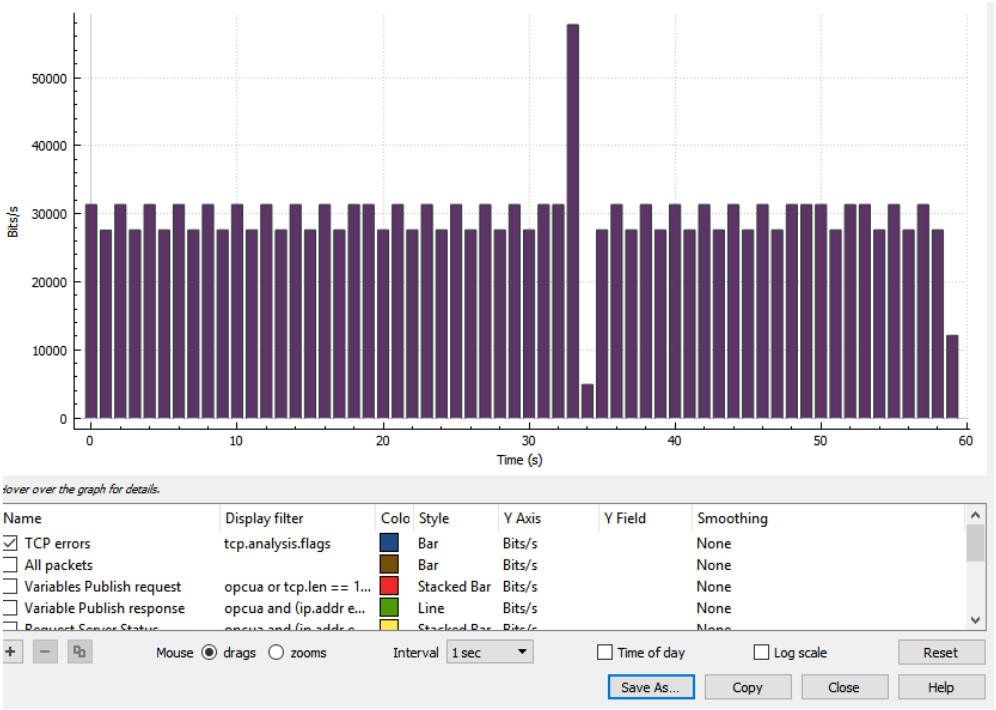

Figura 3.57: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 9a. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	486	448 (92.2%)	N/A
Time span, s	59.000	59.000	N/A
Average pps	8.2	7.6	N/A
Average packet size, B	485.5	511.5	N/A
Bytes	235787	229223 (97.2%)	0
Average bytes/s	3996	3885	N/A
Average bits/s	31 k	31 k	N/A

Figura 3.58: Estadísticas de la conversación OPC UA. Escenario de prueba 9a. Fuente autor.

En cuanto al establecimiento del canal seguro se observa el cambio de algoritmo de Basic128Rsa15 a Basic256 tanto en la petición como en la respuesta del establecimiento.

En comparación de los tiempos de retorno en la lectura de los 100 datos tipo doble no se observa mayor diferencia, en ambos algoritmos se nota un tiempo de viaje de máximo 21 milisegundos, como se observa en la figura 3.59.

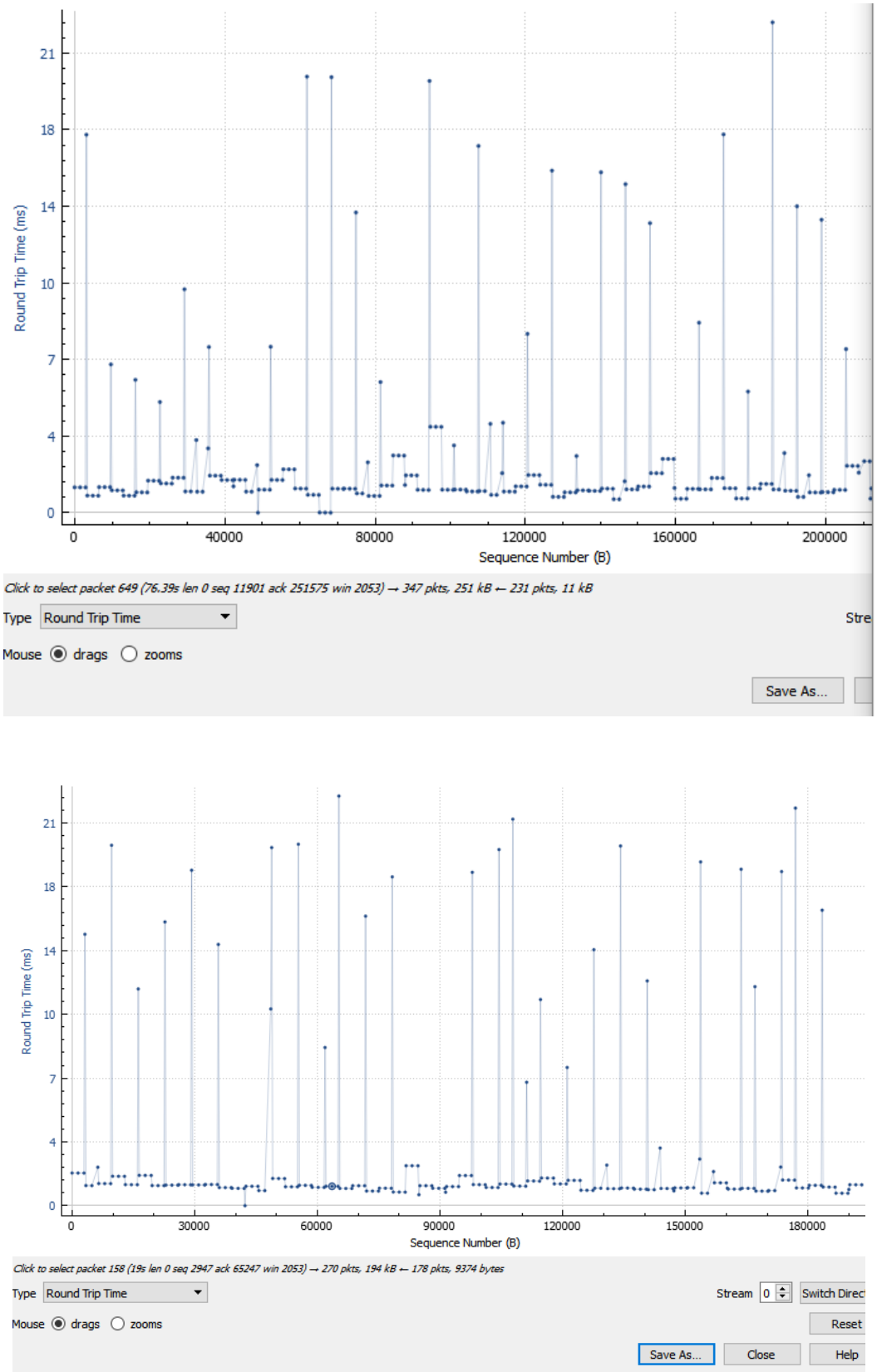

Figura 3.59: Gráfico del tiempo de ida y retorno para el caso de una comunicación de datos firmados con la política de seguridad Basic128Rsa15 (arriba) y Basic256 (abajo). Escenario de prueba 9a. Fuente autor.

3.7.10 Escenario de prueba 10.

3.7.10.1 *Escenario de prueba 10a (Lectura de 100 datos).*

En el siguiente escenario **(Con mensaje firmado y política de seguridad Basic256, usuario anónimo y tiempo de muestreo de 500 milisegundos),** se tiene la lectura de 100 datos tipo doble a 500 milisegundos, en este caso se necesitan 3194 bytes o 3194 x 8 = 25552 bits tomando en cuenta dos mensajes por segundo se tiene 51104 bps, no se observa mucha diferencia con el caso de política de seguridad Basic128Rsa15. En la siguiente figura se muestra la captura de tráfico del escenario de prueba.

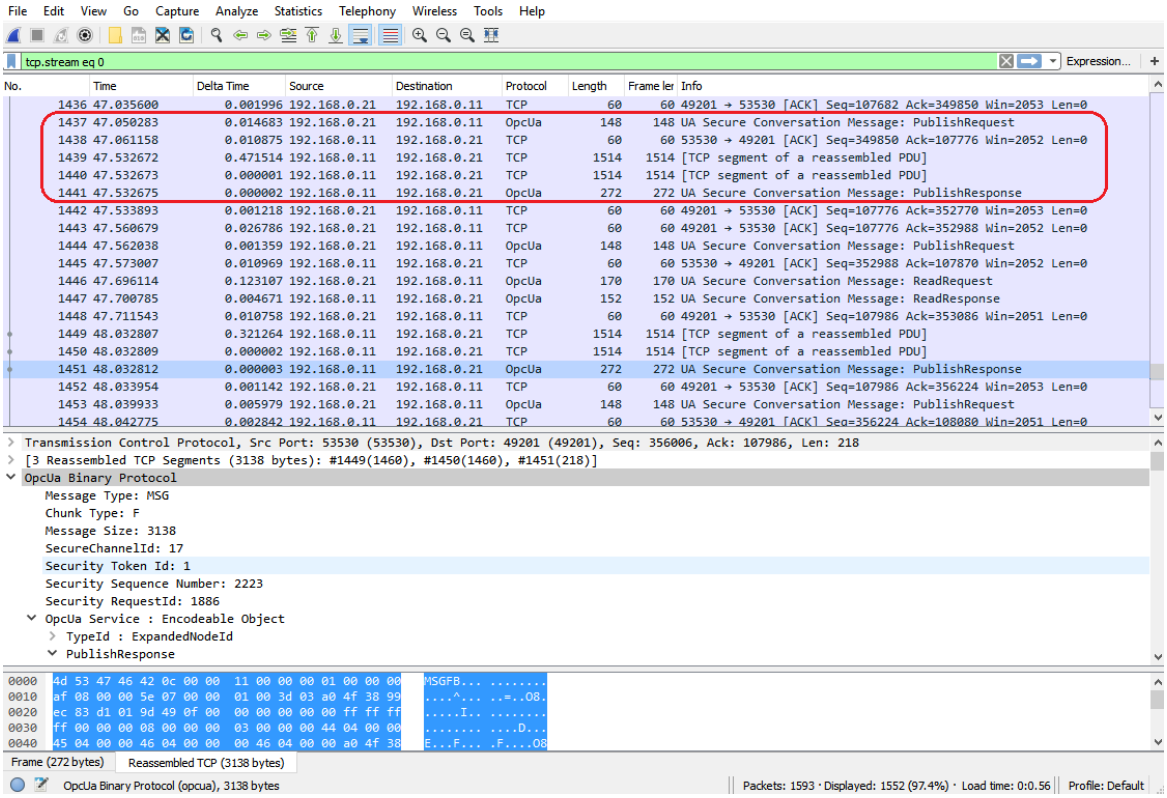

Figura 3.60: Captura el tráfico OPC UA. Escenario de prueba 10a. Fuente autor.

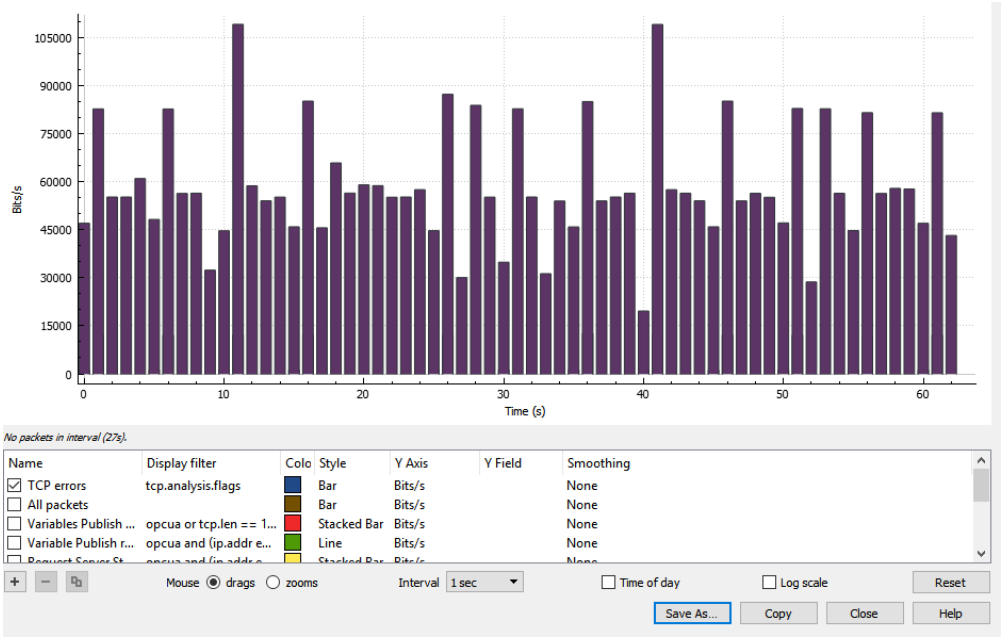

Figura 3.61: Gráfico del uso del ancho de banda del tráfico OPC UA. Escenario de prueba 10a. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	951	951 (100.0%)	N/A
Time span, s	62.982	62.982	N/A
Average pps	15.1	15.1	N/A
Average packet size, B	512.5	512.5	N/A
Bytes	487737	487737 (100.0%)	0
Average bytes/s	7744	7744	N/A
Average bits/s	61 k	61 k	N/A

Figura 3.62: Estadísticas de la conversación OPC UA. Escenario de prueba 10a. Fuente autor.

3.7.11 Escenario de prueba 11.

3.7.11.1 *Escenario de prueba 11a (Lectura de 100 datos).*

En el siguiente escenario **(Con mensaje firmado y política de seguridad Basic256, usuario anónimo y tiempo de muestreo de 100 milisegundos),** se tiene la lectura de 100 datos tipo doble a 100, en este caso se necesitan 255520 bps y se observa que usa los mismos 20 bytes para la firma digital, no existe diferencia en cantidad de bytes con respecto a la política de seguridad Basic128Rsa15. En la siguiente figura 3.63 se muestra la captura de tráfico del escenario de prueba.

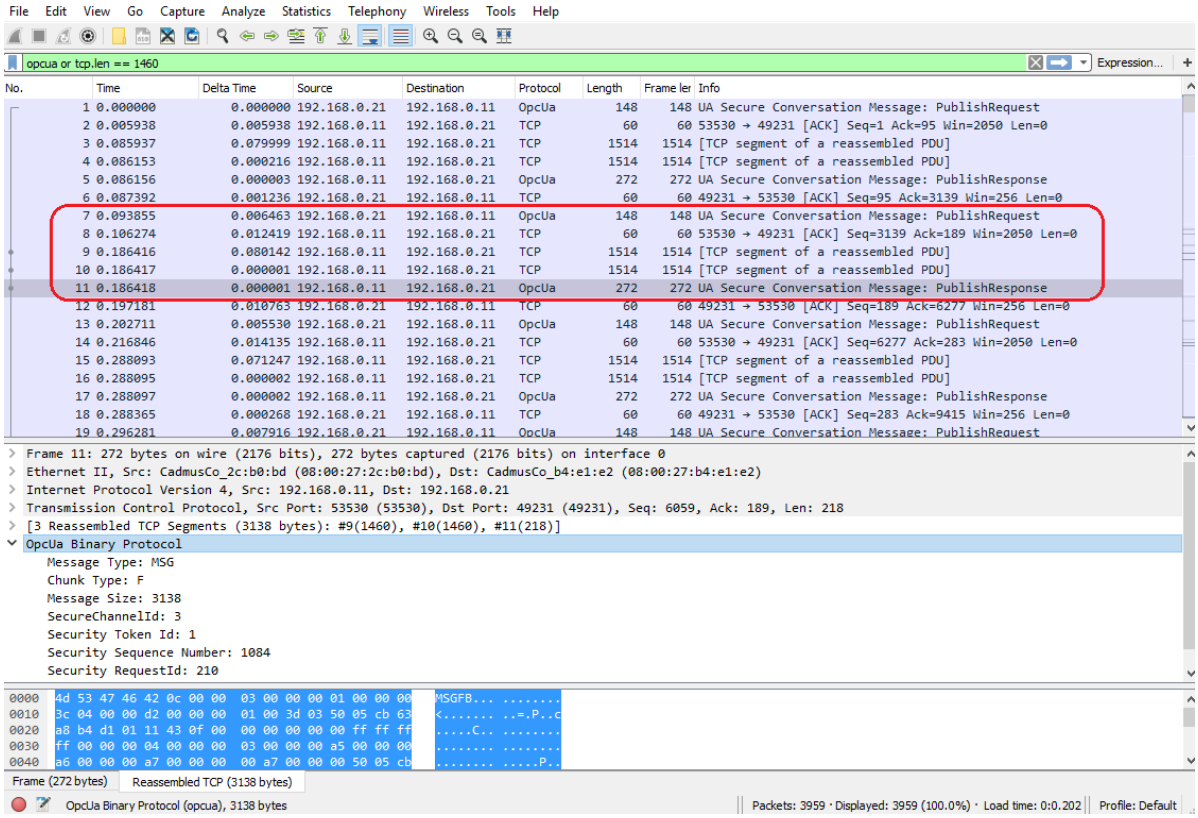

Figura 3.63: Captura el tráfico OPC UA. Escenario de prueba 11a. Fuente autor.

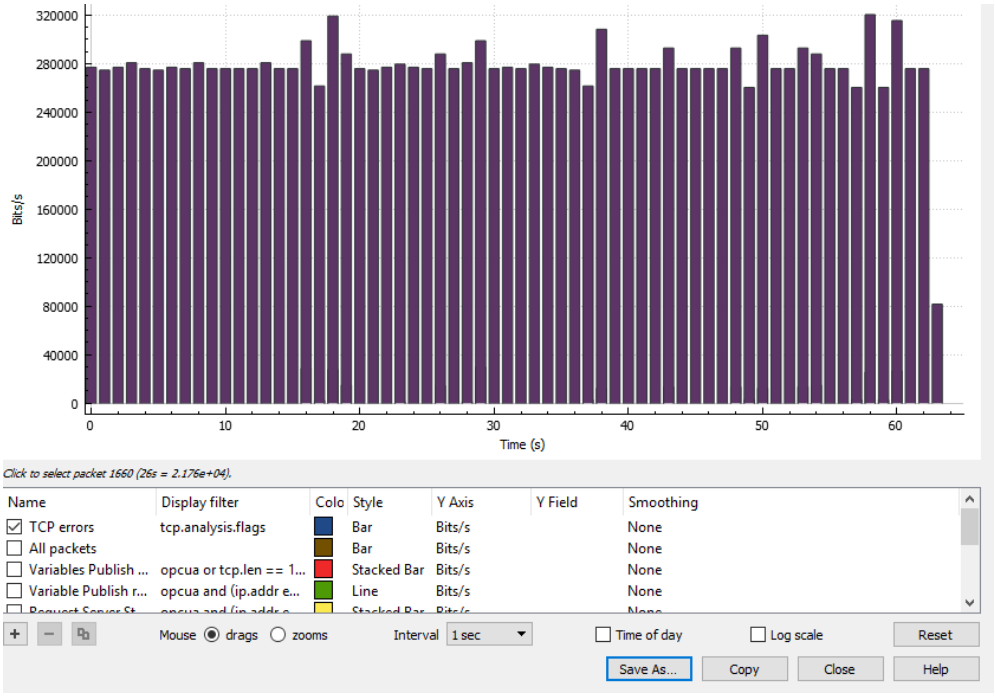

Figura 3.64: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 11a. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	3959	3959 (100.0%)	N/A
Time span, s	63.286	63.286	N/A
Average pps	62.6	62.6	N/A
Average packet size, B	582.5	582.5	N/A
Bytes	2305285	2305285 (100.0%)	0
Average bytes/s	36 k	36 k	N/A
Average bits/s	291 k	291 k	N/A

Figura 3.65: Estadísticas de la conversación OPC UA. Escenario de prueba 11a. Fuente autor.

3.7.12 Escenario de prueba 12.

3.7.12.1 *Escenario de prueba 12a (Lectura de 100 datos).*

En este escenario de prueba se realiza la conexión **con mensaje firmado y política de seguridad Basic256SHA256, usuario anónimo y tiempo de muestreo de 100 milisegundos**. Al igual que en las políticas de seguridad Basic128Rsa15 y Basic256, no se observa diferencia significativa en la cantidad de bytes transmitidos con la política de seguridad Basic256Sha256, como se puede observar en la figura 3.66.

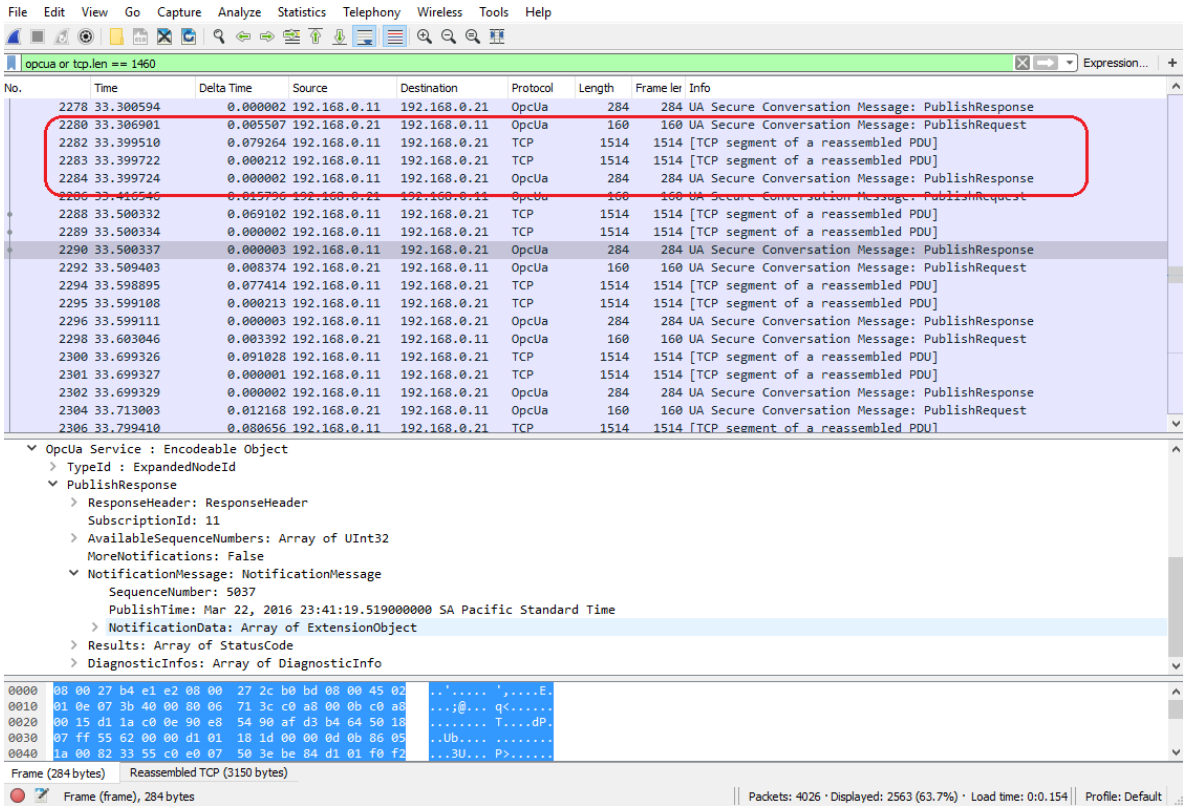

Figura 3.66: Captura el tráfico OPC UA. Escenario de prueba 12a. Fuente autor.

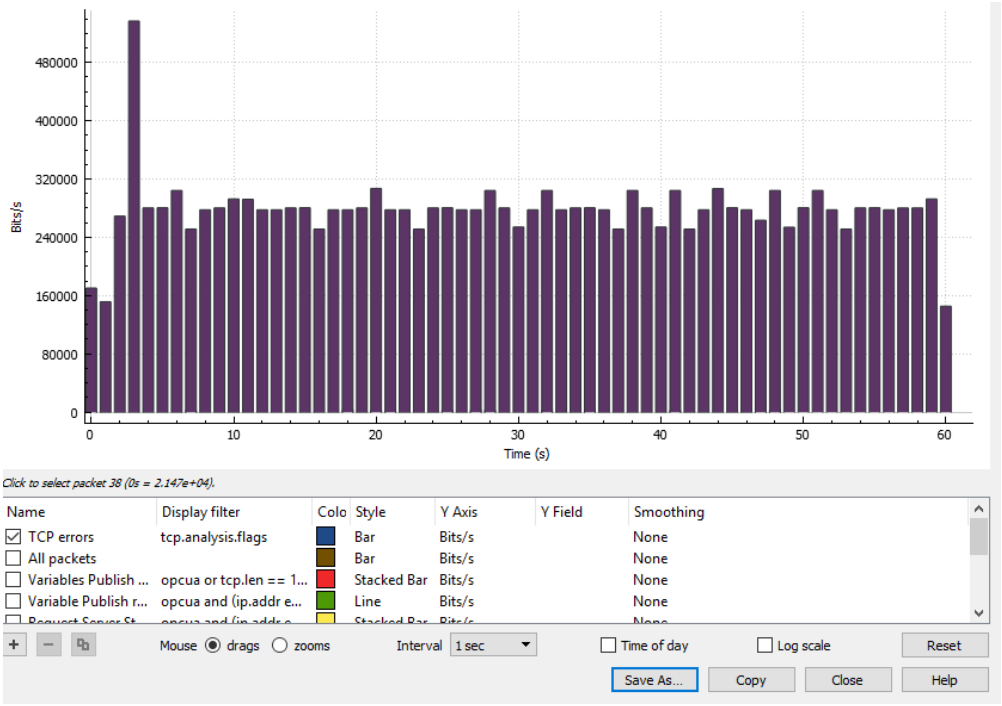

Figura 3.67: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 12a. Wireshark. Captura de pantalla-Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	4026	2563 (63.7%)	N/A
Time span, s	60.435	60.434	N/A
Average pps	66.6	42.4	N/A
Average packet size, B	548.5	825.5	N/A
Bytes	2208001	2114716 (95.8%)	0
Average bytes/s	36 k	34 k	N/A
Average bits/s	292 k	279 k	N/A

Figura 3.68: Estadísticas de la conversación OPC UA. Escenario de prueba 12a. Fuente autor.

3.7.13 Escenario de prueba 13.

3.7.13.1 *Escenario de prueba 13a (Lectura de 100 datos).*

En el siguiente escenario de **prueba (Con mensaje firmado y cifrado, política de seguridad Basic128RSA15, usuario anónimo y tiempo de muestreo de 1 segundo)**, se habilita el modo de seguridad para firmado y cifrado digital con la política Basic128RSA15, debido al cifrado que se realiza sobre el mensaje solo se pueden observar las cabeceras OPC UA. A diferencia del modo de seguridad de firmado digital, se observa

un aumento de 10 bytes en el cuerpo del mensaje y un aumento del tiempo de ida y retorno como se puede observar en la figura 3.69 y 3.70 respectivamente. Debido a este aumento de 10 bytes se denota un aumento en el ancho de banda utilizado que llega a los 30000 bps, teniendo en cuenta que en el modo de seguridad únicamente firmado se llegaba a aproximadamente los 28000 bps.

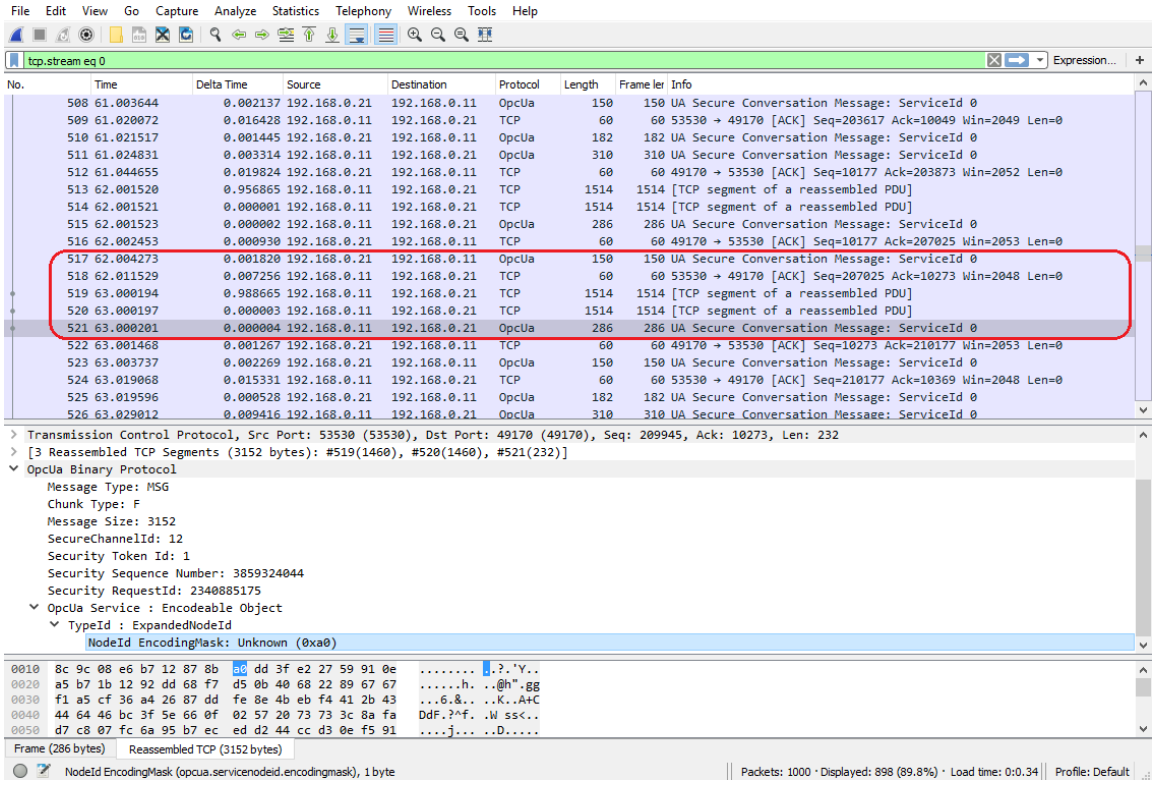

Figura 3.69: Captura el tráfico OPC UA. Escenario de prueba 13a. Fuente autor.

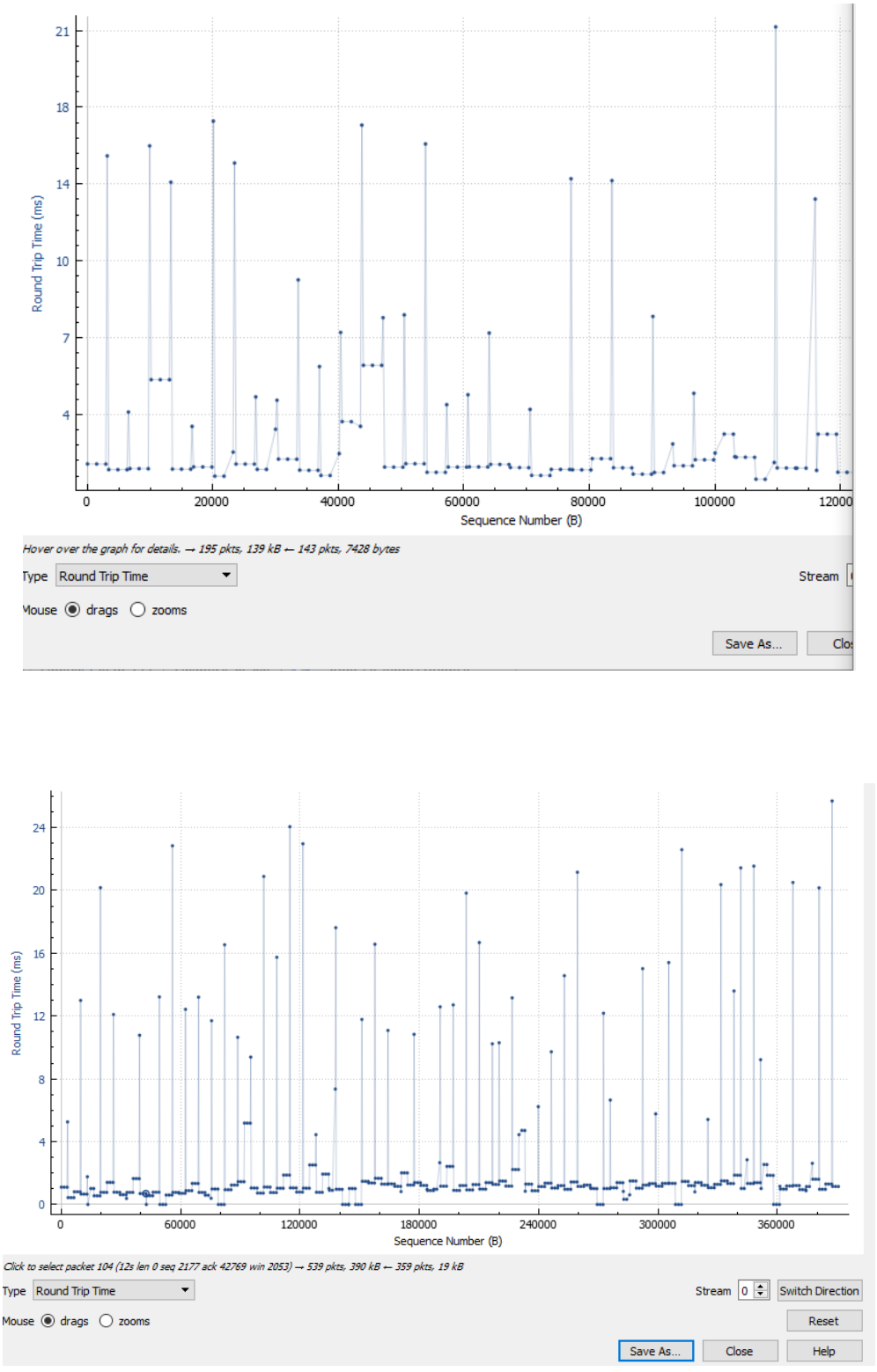

Figura 3.70: Gráfico del tiempo de ida y retorno para el caso de una comunicación de datos firmados con la política de seguridad Basic128Rsa15 con método de seguridad = firmado (arriba) y método de seguridad = firmado y cifrado (abajo). Escenario de prueba 13a. Fuente autor.

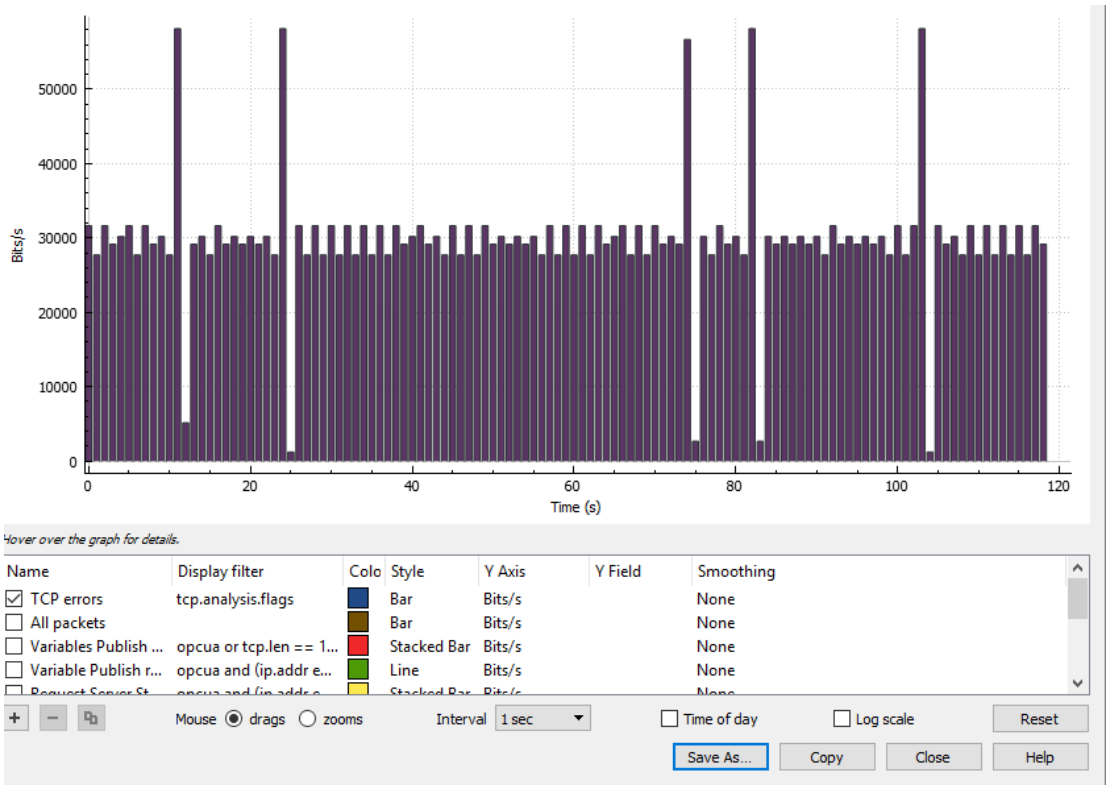

Figura 3.71: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 13a. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	1000	898 (89.8%)	N/A
Time span, s	118.998	118.998	N/A
Average pps	8.4	7.5	N/A
Average packet size, B	473.5	513.5	N/A
Bytes	473673	460722 (97.3%)	0
Average bytes/s	3980	3871	N/A
Average bits/s	31 k	30 k	N/A

Figura 3.72: Estadísticas de la conversación OPC UA. Escenario de prueba 13a. Fuente autor.

3.7.14 Escenario de prueba 14.

3.7.14.1 *Escenario de prueba 14a (Lectura de 100 datos).*

En el siguiente escenario de prueba se realiza la conexión **con mensaje firmado y cifrado, política de seguridad Basic128RSA15, usuario anónimo y tiempo de muestreo de 500 milisegundos**. De igual forma que en el caso anterior se observa un aumento de 10 bytes en el cuerpo del mensaje obteniéndose una diferencia no muy significativa. El ancho de banda utilizado es el doble del muestreo a 1 segundo por lo que se tendrá un ancho de banda que rodea los 60000 bps, como se muestra en las siguientes figuras.

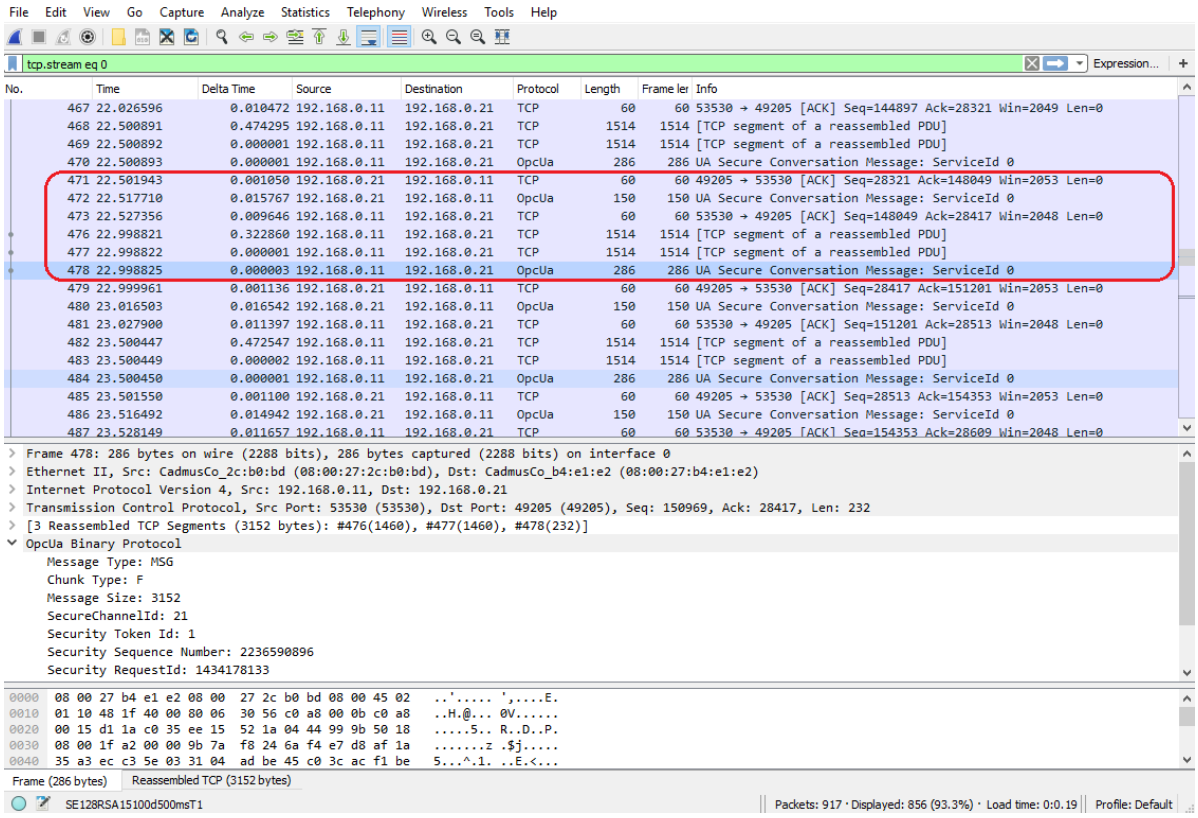

Figura 3.73: Captura el tráfico OPC UA. Escenario de prueba 14a. Fuente autor.

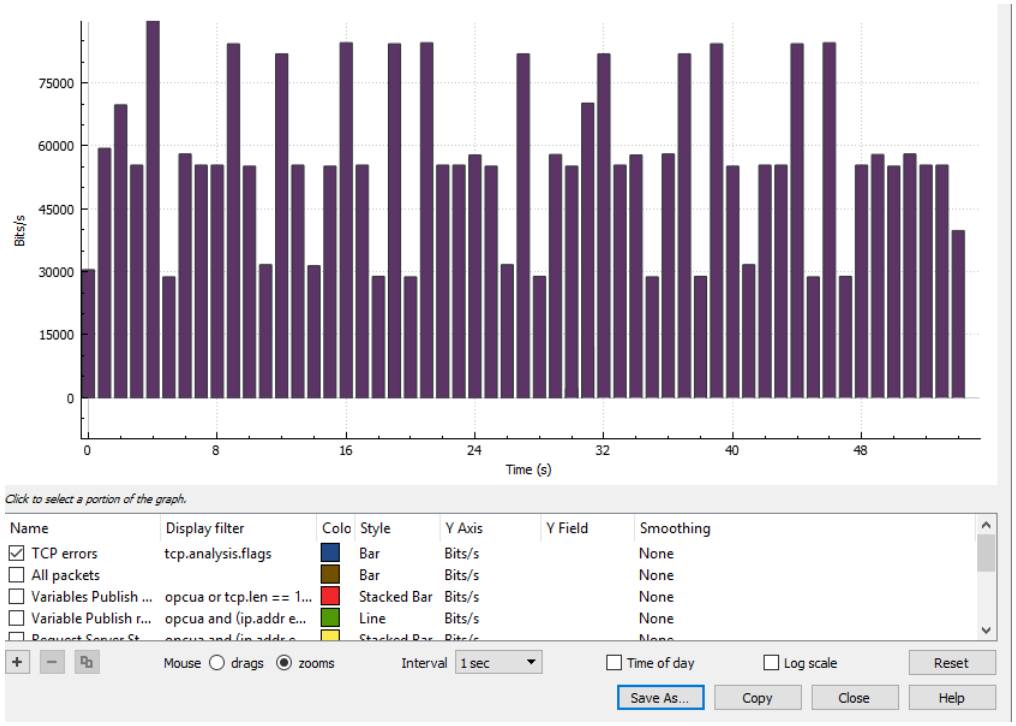

Figura 3.74: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 14a. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	917	856 (93.3%)	N/A
Time span, s	54.500	54.500	N/A
Average pps	16.8	15.7	N/A
Average packet size, B	481.5	507.5	N/A
Bytes	441156	434492 (98.5%)	0
Average bytes/s	8094	7972	N/A
Average bits/s	64 k	63 k	N/A

Figura 3.75: Estadísticas de la conversación OPC UA. Escenario de prueba 14a. Fuente autor.

3.7.1 Escenario de prueba 15.

3.7.1.1 *Escenario de prueba 15a (Lectura de 100 datos).*

En el siguiente escenario de prueba se realiza la conexión **con mensaje firmado y cifrado, política de seguridad Basic128RSA15, usuario anónimo y tiempo de muestreo de 100 milisegundos**. Al igual que en el caso anterior se tiene una diferencia de 10 bytes por cada mensaje de lectura de los 100 datos de tipo doble, y al tener 10

menajes por segundo se tiene una diferencia de 100 bytes por segundo, esta diferencia se puede observar en las siguientes figuras.

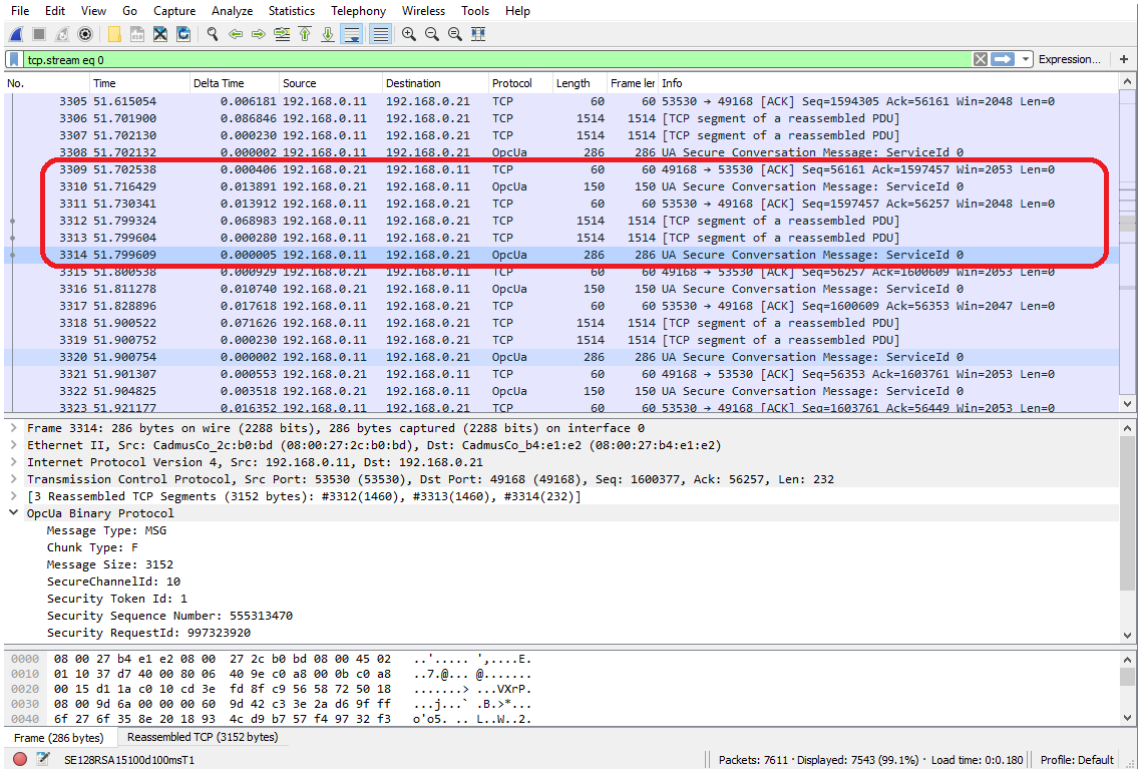

Figura 3.76: Captura el tráfico OPC UA. Escenario de prueba 15a. Fuente autor.

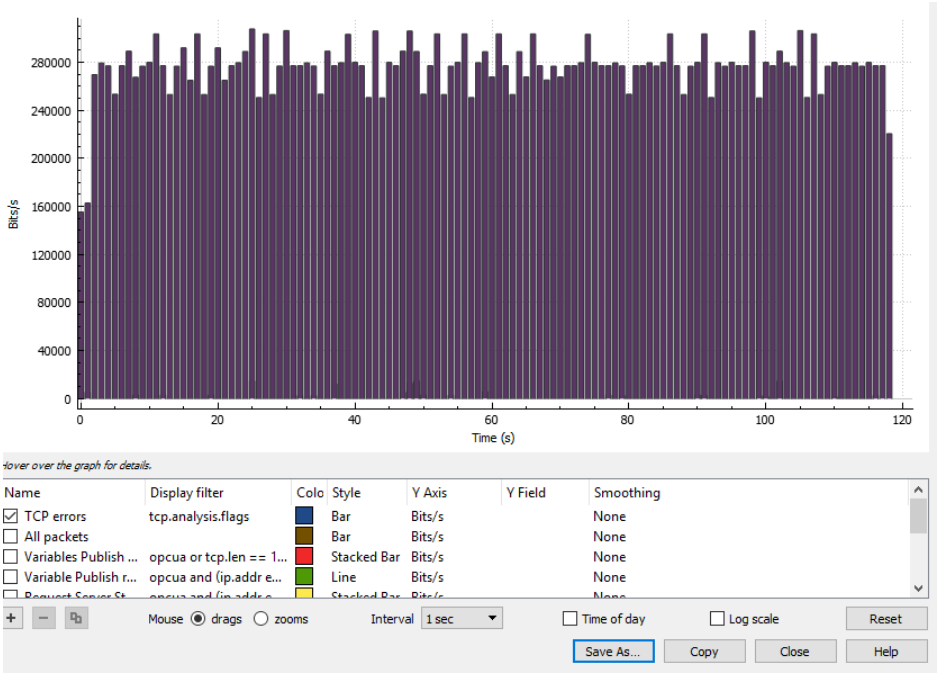

Figura 3.77: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 15a. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	7611	7543 (99.1%)	N/A
Time span, s	118.701	118.701	N/A
Average pps	64.1	63.5	N/A
Average packet size, B	562.5	566.5	N/A
Bytes	4280904	4270984 (99.8%)	0
Average bytes/s	36 k	35 k	N/A
Average bits/s	288 k	287 k	N/A

Figura 3.78: Estadísticas de la conversación OPC UA. Escenario de prueba 15a. Wireshark. Captura de pantalla-Fuente el autor.

3.7.2 Escenario de prueba 16.

3.7.2.1 *Escenario de prueba 16a (lectura 100 datos).*

En el siguiente escenario de prueba se realiza la conexión **con mensaje firmado y cifrado, política de seguridad Basic256, usuario anónimo y tiempo de muestreo de 100 milisegundos**. En comparación con la política de seguridad Basic128Rsa15, a nivel de longitud de datos en los mensajes, no se observa mucha diferencia en el uso del ancho de banda que en ambos casos bordea los 280000 bps. En las siguientes figuras se observa la captura del tráfico y el grafico del uso de ancho de banda para el escenario de prueba 16.

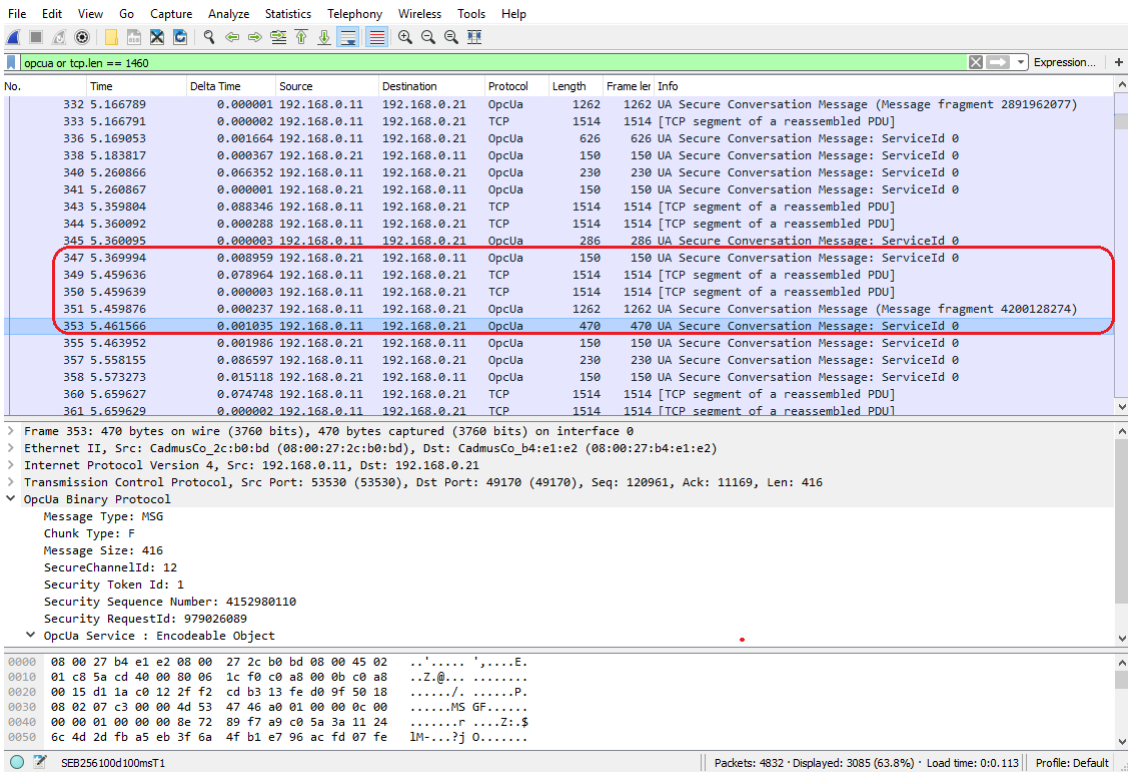

Figura 3.79: Captura el tráfico OPC UA. Escenario de prueba 16a. Fuente autor.

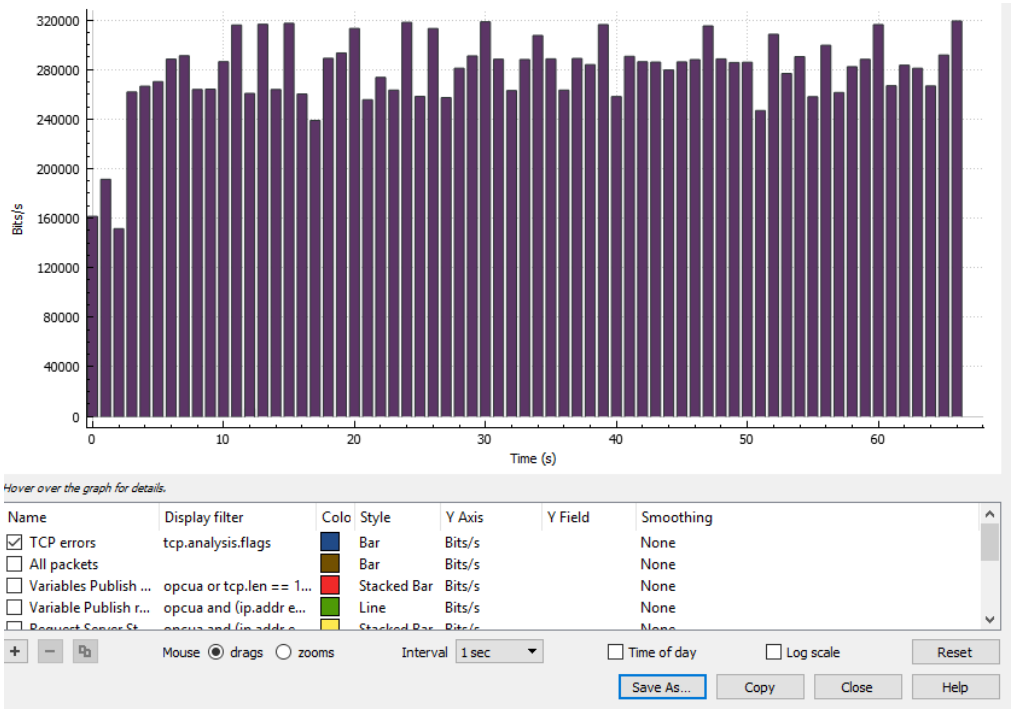

Figura 3.80: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 16a. Fuente autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	4832	3085 (63.8%)	N/A
Time span, s	66.964	66.963	N/A
Average pps	72.2	46.1	N/A
Average packet size, B	505.5	756.5	N/A
Bytes	2444300	2333726 (95.5%)	0
Average bytes/s	36 k	34 k	N/A
Average bits/s	292 k	278 k	N/A

Figura 3.81: Estadísticas de la conversación OPC UA. Escenario de prueba 16a. Fuente autor.

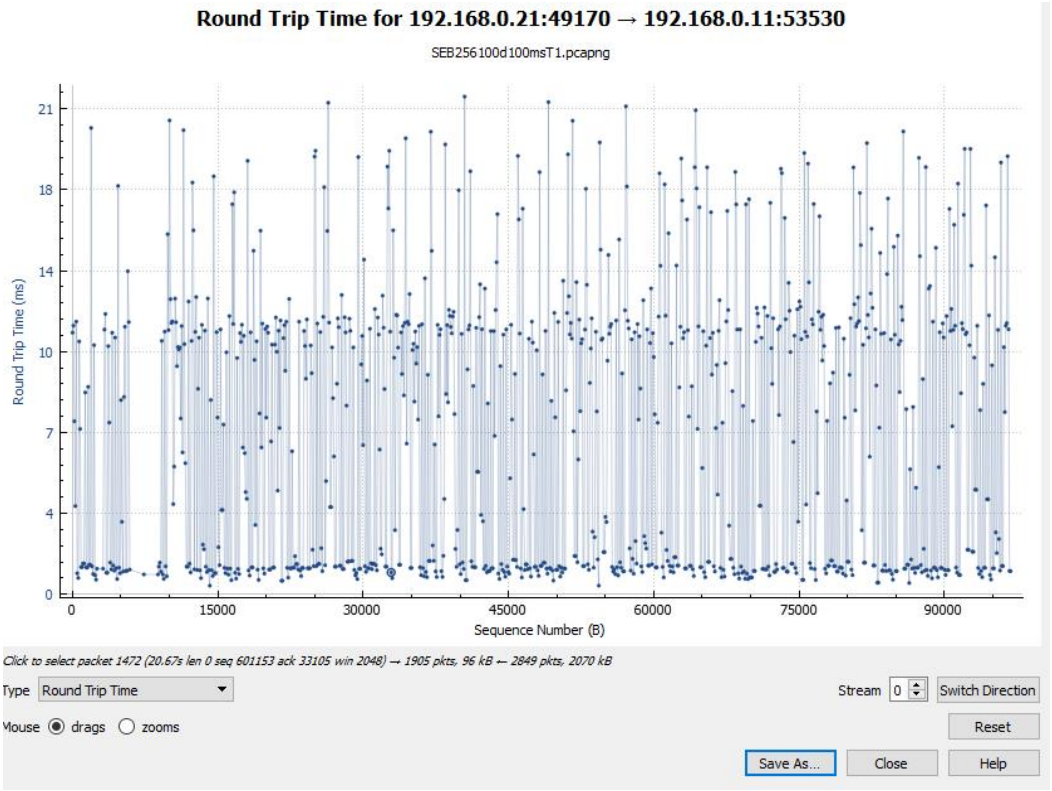

Figura 3.82: Gráfico del tiempo de ida y retorno para el escenario de prueba 16a. Fuente autor.

3.7.3 Escenario de prueba 17.

3.7.3.1 *Escenario de prueba 17a (Lectura de 100 datos).*

En el siguiente escenario de prueba se realiza la conexión **con mensaje firmado y cifrado, política de seguridad Basic256SHA256, usuario anónimo y tiempo de muestreo de 100 milisegundos**. Como último caso se tiene la lectura de 100 datos de tipo doble a 100 milisegundos con política de seguridad Basic256SHA256 y modo de seguridad = firmado y encriptado, en el cual tampoco se observa diferencia significativa en cuanto a longitud extra de bytes en relación con la lectura de igual datos y política de seguridad Basic256.

En cuanto al tiempo de ida y retronó si se denota diferencia en comparación a la política de seguridad Basic256 en donde se tenían tiempos picos de 21 milisegundos y en el caso de la política de seguridad Basic256SHA256 se tienen picos de 60 milisegundos, como se puede observar en las siguientes figuras.

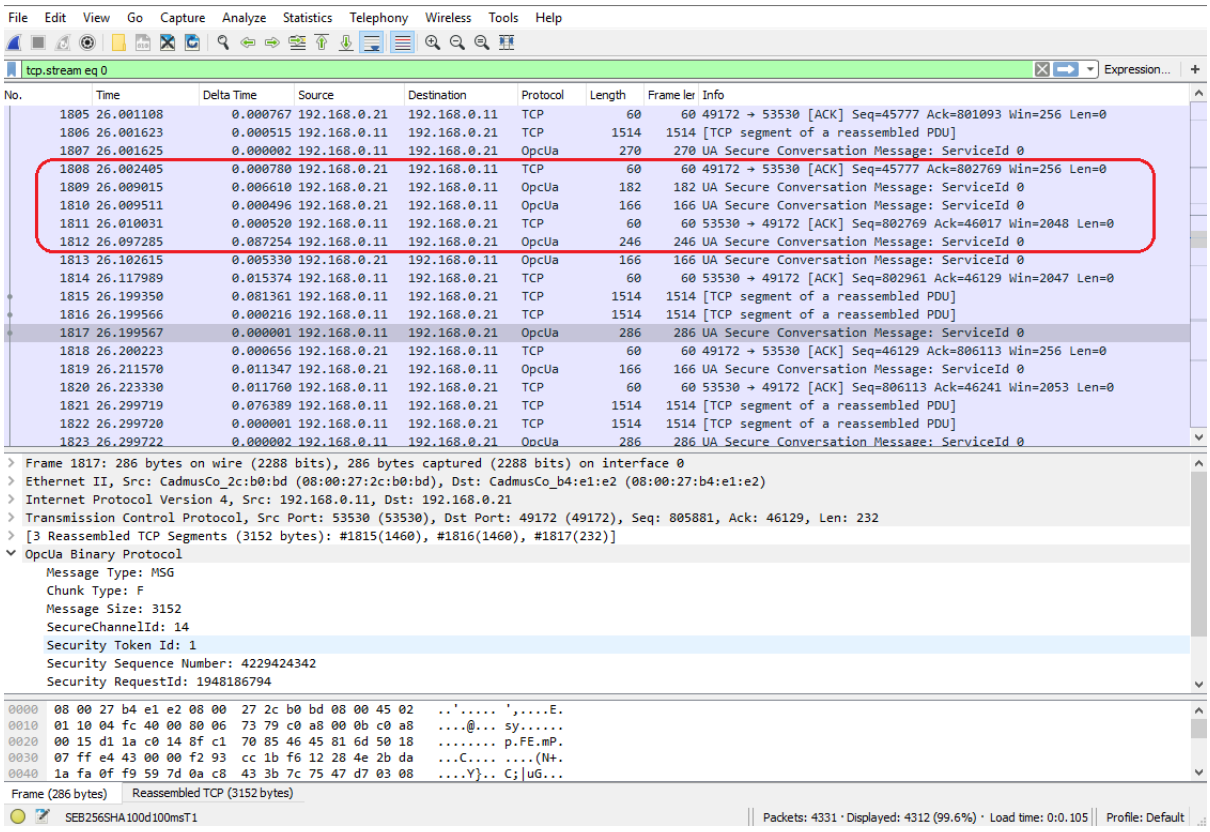

Figura 3.83: Captura el tráfico OPC UA. Escenario de prueba 17a. Wireshark. Captura de pantalla-Fuente el autor.

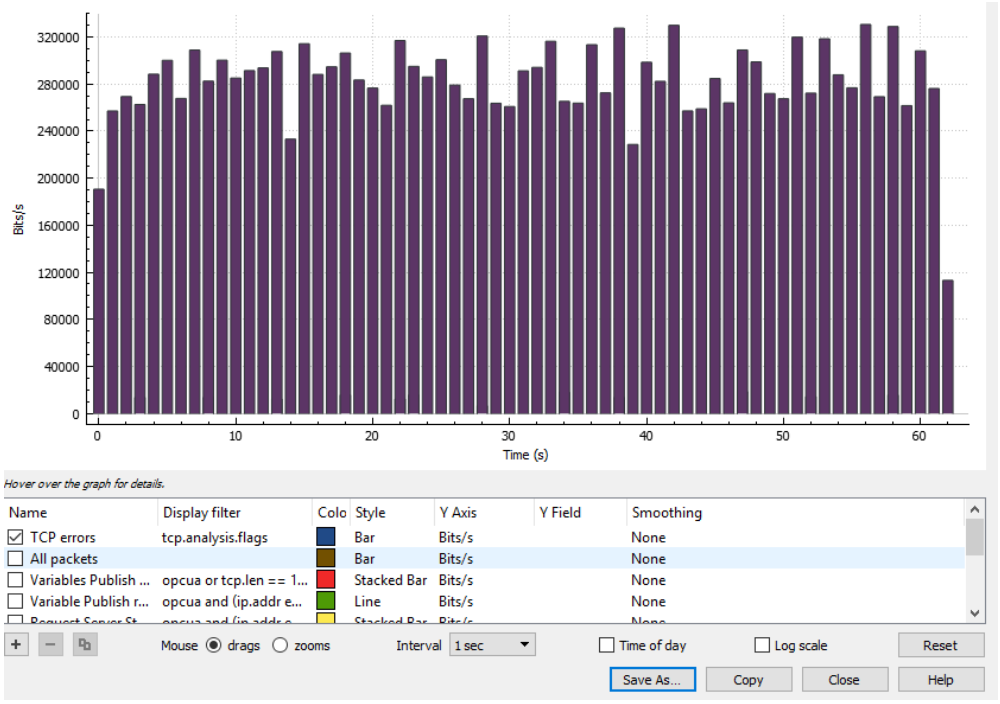

Figura 3.84: Gráfico del uso del ancho de banda del tráfico OPC UA. Escenario de prueba 17a. Wireshark. Captura de pantalla-Fuente el autor.

Statistics

Measurement	Captured	Displayed	Marked
Packets	4331	4312 (99.6%)	N/A
Time span, s	62.305	62.305	N/A
Average pps	69.5	69.2	N/A
Average packet size, B	535.5	536.5	N/A
Bytes	2318601	2315354 (99.9%)	0
Average bytes/s	37 k	37 k	N/A
Average bits/s	297 k	297 k	N/A

Figura 3.85: Estadísticas de la conversación OPC UA. Escenario de prueba 17a. Wireshark. Captura de pantalla-Fuente el autor.

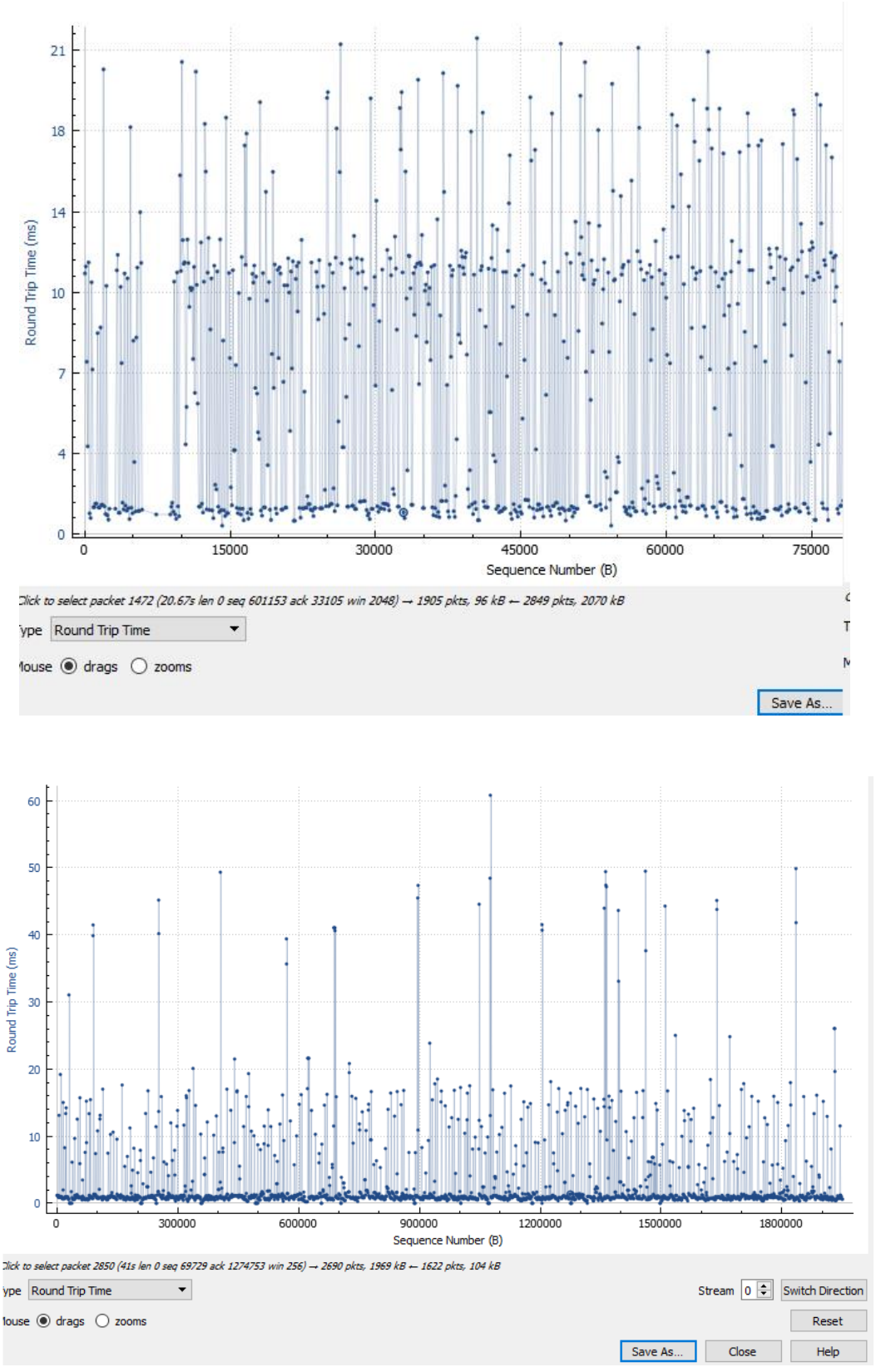

Figura 3.86: Gráfico del tiempo de ida y retorno para el caso de una comunicación de datos firmados-cifrados y política de seguridad Basic256 (arriba) vs política de seguridad Basic256SHA256 (abajo). Escenario de prueba 17a. Fuente autor.

3.8 ANALISIS COMPARATIVO DE LOS ESCENARIOS DE PRUEBA.

3.8.1 Análisis comparativo de los escenarios de prueba con lectura de datos sin firmado digital y sin ningún cifrado.

En la siguiente Tabla 3.5 se puede observar los escenarios de pruebas en los cuales no se tiene firma digital ni cifrado en los datos, en la cual se tienen los campos del ancho de banda de petición desde el cliente al servidor OPC UA y el ancho de banda de la respuesta. Mientras el número de datos es mayor existe más segmentación y el ancho de banda aumenta. De igual forma cuando se aumenta el tiempo de muestreo, existe mayor número de respuestas y el ancho de banda aumenta. Al utilizar autenticación de usuario con contraseña o con certificado no se altera el ancho de banda de lectura de datos pero si se altera en la activación de la sesión en donde se trasmite la contraseña o el certificado.

Tabla 3.5: Tabla comparativa de los escenarios de prueba con lectura de datos sin firmado digital y sin ningún cifrado. Fuente el autor.

Escenario de prueba	Modo de seguridad	Política de seguridad	Autenticación de usuario	Tiempo de muestreo	Numero de datos	Numero de segmentos del paquete	Número de peticiones por segundo	Numero de bytes por datos	Ancho de banda (Petición lectura de datos)	Ancho de banda (Respuesta lectura de datos)	Ancho de banda (Petición estado del servidor)	Ancho de banda (Respuesta estado del servidor)	Ancho de banda total (Petición datos y estado servidor)	Ancho de banda total (Respuesta datos y estado servidor)	Observaciones
				(ms)	(Tipo doble)				(bps)	(bps)	(bps)	(bps)	(bps)	(bps)	
Prueba 1	Ninguno	Ninguno	Anónimo	1000	5	1	1	150	1504	3088	1680	2680	3184	5768	Contraseña o certificado enviado unicamente en la activacion de la sesion, mas no en los datos.
				1000	11	1	1	330	1504	4528	1680	2680	3184	7208	
				1000	51	2	1	1530	1504	14560	1680	2680	3184	17240	
				1000	1000	21	1	30000	1504	251104	1680	2680	3184	253784	
Prueba 2	Ninguno	Ninguno	Anónimo	500	1000	21	2	30000	3008	502208	1680	2680	4688	504888	
Prueba 3	Ninguno	Ninguno	Anónimo	100	1000	21	10	30000	15040	2511040	1680	2680	16720	2513720	
Prueba 4	Ninguno	Ninguno	Contraseña	100	1000	21	10	30000	15040	2511040	1680	2680	16720	2513720	
Prueba 5	Ninguno	Ninguno	Certificado	100	1000	21	10	30000	15040	2511040	1680	2680	16720	2513720	

3.8.2 Análisis comparativo de los escenarios de prueba con lectura de datos con firmado digital y sin cifrado.

En la siguiente Tabla 3.6 se pueden observar los escenarios de pruebas en los cuales se tiene firma digital y sin cifrado en los datos, Debido a la firma digital se envían 20 bytes extras en cada petición y respuesta de los datos por lo tanto el ancho de banda aumenta inclusive si se usa segmentación, cada pedazo del paquete tiene la firma digital al final de los datos.

Tabla 3.6: Tabla comparativa de los escenarios de prueba con lectura de datos con firmado digital y sin cifrado. Fuente el autor.

Escenario de prueba	Modo de seguridad	Política de seguridad	Autenticación de usuario	Tiempo de muestreo	Número de datos	Número de paquetes	Número de peticiones por segundo	Número de bytes por datos	Ancho de banda (Petición lectura de datos)	Ancho de banda (Respuesta lectura de datos)	Ancho de banda (Petición estado del servidor)	Ancho de banda (Respuesta estado del servidor)	Ancho de banda total (Petición datos y estado servidor)	Ancho de banda total (Respuesta datos y estado servidor)	Observaciones
				(ms)	(Tipo doble)				(bps)	(bps)	(bps)	(bps)	(bps)	(bps)	
Prueba 6	Mensaje Firmado	Basic128RSA15	Anónimo	1000	6	1	1	180	1664	3488	1840	2840	3504	6328	Difiere en 20 bytes en la petición y respuesta de lectura de datos debido a la firma digital.
				1000	100	3	1	3000	1664	26912	1840	2840	3504	29752	
Prueba 7	Mensaje Firmado	Basic128RSA15	Anónimo	500	100	3	2	3000	3328	53824	1840	2840	5168	56664	
Prueba 8	Mensaje Firmado	Basic128RSA16	Anónimo	100	100	3	10	3000	16640	269120	1840	2840	18480	271960	
Prueba 9	Mensaje Firmado	Basic256	Anónimo	1000	100	3	1	3000	1664	26912	1840	2840	3504	29752	
Prueba 10	Mensaje Firmado	Basic256	Anónimo	500	100	3	2	3000	3328	53824	1840	2840	5168	56664	
Prueba 11	Mensaje Firmado	Basic256	Anónimo	100	100	3	10	3000	16640	269120	1840	2840	18480	271960	
Prueba 12	Mensaje Firmado	Basic256SHA256	Anónimo	100	100	3	10	3000	17600	269760	1840	2840	19440	272600	

3.8.3 Análisis comparativo de los escenarios de prueba con lectura de datos con firmado digital y cifrado.

En la siguiente Tabla 3.7 se pueden observar los escenarios de pruebas en los cuales se tiene firma digital y cifrado en los datos, No se observa un aumento significativo en el ancho de banda utilizado en comparación a la lectura de datos solo con firma digital. Una de las diferencias notables se observan en el tiempo de ida y retorno de los mensajes de los datos en donde se puede observar un aumento del doble de ida y retorno probablemente por el procesamiento de cifrado y descifrado.

Tabla 3.7: Tabla comparativa de los escenarios de prueba con lectura de datos con firmado digital y cifrado. Fuente el autor.

Escenario de prueba	Modo de seguridad	Política de seguridad	Autenticación de usuario	Tiempo de muestreo	Número de datos	Número de paquetes	Número de peticiones por segundo	Número de bytes por datos	Ancho de banda (Petición lectura de datos)	Ancho de banda (Respuesta lectura de datos)	Ancho de banda (Petición estado del servidor)	Ancho de banda (Respuesta estado del servidor)	Ancho de banda total (Petición datos y estado servidor)	Ancho de banda total (Respuesta datos y estado servidor)	Observaciones
				(ms)	(Tipo doble)				(bps)	(bps)	(bps)	(bps)	(bps)	(bps)	
Prueba 13	Mensaje Firmado y Cifrado	Basic128RSA15	Anónimo	1000	100	3	1	3000	1680	26992	1840	2840	3520	29832	Aumento del tiempo de ida y retorno de los datos debido al proceso de cifrado
Prueba 14	Mensaje Firmado y Cifrado	Basic128RSA15	Anónimo	500	100	3	2	3000	3360	53984	1840	2840	5200	56824	
Prueba 15	Mensaje Firmado y Cifrado	Basic128RSA15	Anónimo	100	100	3	10	3000	16800	269920	1840	2840	18640	272760	
Prueba 16	Mensaje Firmado y Cifrado	Basic256	Anónimo	100	100	3	10	3000	16800	269920	1840	2840	18640	272760	
Prueba 17	Mensaje Firmado y Cifrado	Basic128RSA15	Anónimo	100	100	3	10	3000	16800	269920	1840	2840	18640	272760	

3.8.4 Ecuación de estimación del ancho de banda para los escenarios de prueba de lectura de datos con y sin firmado digital.

De los escenarios de prueba se puede obtener valores para obtener una ecuación de la estimación del ancho de banda. Se tienen los siguientes datos:

Nd: Numero de datos a leer mediante OPC UA.
Tbtd: Tamaño en bytes por cada dato de tipo doble = 30 bytes.
Tbfd: Tamaño en bytes si se aplica firmado digital = 20 bytes.
Fdh: Firmado digital, tiene un valor de 1 si se habilita.
Tbmtu: Tamaño en bytes del MTU utilizado = 1514 bytes.
Tbmtusc: Tamaño en bytes del MTU sin cabecera Ethernet e IP = 1460 bytes
Tbcopcua: Tamaño en bytes de la cabecera OPC UA = 122 bytes.
Tbceti: Tamaño en bytes de la cabecera Ethernet, tcp e ip = 54 bytes.
Tbpd: Tamaño en bytes de la petición de datos OPC UA = 128 bytes.
Tbpdfd: Tamaño en bytes de la petición de datos OPC UA con firma digital.
Tbpe: Tamaño en bytes de la petición del estado del servidor OPC UA = 150 bytes
Tbpefd: Tamaño en bytes de la petición del estado del servidor con firma digital.
Tbre: Tamaño en bytes de la respuesta del estado del servidor OPC UA = 275 bytes
Tbrefd: Tamaño en bytes de la respuesta del estado del servidor con firma digital.
Tbackes: Tamaño en bytes del ACK del estado del servidor = 120 bytes.
Tbackrd: Tamaño en bytes del ACK de la respuesta de datos = 120 bytes.
Tbrd: Tamaño en bytes de la respuesta de datos OPC UA.
Tbrdfd: Tamaño en bytes de la respuesta de datos OPC UA con firma digital.
Tm: Tiempo de muestreo en milisegundos.

Para obtener el número total de bytes de la respuesta de los datos se debe multiplicar la cantidad de datos a leer por 30 bytes de cada tipo doble y adicionar el tamaño de la cabecera OPC UA y adicionar 20 bytes si se utiliza firmado digital.

Tbdopcuafd: Tamaño en bytes de los datos con cabecera OPC UA y firmado digital.

$$Tbdopcuafd = [\,(Nd\ x\ Tbtd) + Tbcopcua + (Fdh * 20)]$$

(3-1)

Adicionalmente se debe calcular el número de segmentos utilizados por la cantidad de datos a leer.

Nsgm: Numero de segmentos del mensaje OPC UA.

$$Nsgm = 1 + COCIENTE\left[\frac{Tbdopcuafd}{Tbmtusc}\right]$$

(3-2)

Ahora se puede calcular el tamaño de los datos a leer considerando los segmentos con su respectiva cabecera Ethernet, ip y tcp.

$$Tbrdfd = Tbdopcufd + (Nsgm * Tbceti).$$

(3-3)

Luego se considera la firma digital en los términos anteriores mediante la inserción de la variable de habilitación de firma digital.

$$Tbpdfd = Tbpd + (Fd * 20)$$

(3-4)

$$Tbpefd = Tbpe + (Fd * 20)$$

(3-5)

$$Tbrefd = Tbre + (Fd * 20)$$

(3-6)

Por último se considera el tiempo de muestreo en el cálculo del ancho de banda total, como se indica en la siguiente formula.

ABtotbps: Ancho de banda total en bits por segundo.

$$ABtotbps = (Tbpefd + Tbrefd + ((Tbpdfd + Tbrdfd) * (1000/Tm)) + Tbackes + ((Tbackrd * (1000/Tm))) * 8$$

(3-7)

En la siguiente tabla se muestra un ejemplo de la estimación del ancho de banda utilizado por una lectura de 1000 datos de tipo doble con firmado digital con tiempo de muestreo de 100 milisegundos.

Tabla 3.8: Ejemplo del cálculo de la estimación del ancho de banda para la lectura de 1000 datos.

		Unidad	**Símbolo**
Número de Datos tipo doble	1000		Nd
Tiempo de muestreo	100	milisegundos	Tm
Tamaño MTU	1514	bytes	Tbmtu
Tamaño MTU sin cabecera ETH, TCP, IP	1460	bytes	Tbmtusc
Firmado Digital habilitado	1		Fdh
Cabecera OP CUA	122	bytes	Tbcopcua
Cabecera Ethernet, TCP, IP	54	bytes	Tbcetip

		Unidad	Simbolo
Tamaño en bytes por dato	30	bytes	Tbtd
Tamaño de datos	30000	bytes	
Tamaño de datos mas cabecera OPC UA	30122	bytes	Tbdopcua
Tamaño de datos con cabecera OPC UA y firma digital	30142	bytes	Tbdopcuafd

Segmentos	21	Nsgm

Sin cabecera ETH, TCP, IP	Tamaño segmentos -1	29200	bytes
	Tamaño segmento final	942	bytes
	Tamaño segmentos	30142	bytes

Con cabecera ETH, TCP, IP	Tamaño segmentos -1	30280	bytes			
	Tamaño segmento final	996	bytes			
	Tamaño segmentos	31276	bytes	31276	bytes	Tbrdfd

Tamaño en bytes de petición de datos	128	bytes	Tbpd
Tbpd con bytes por firma digital	148	bytes	Tbpdfd
Tamaño en bytes de respuesta de datos	31276	bytes	Tbrdfd
Tamaño en bytes de petición de estado servidor	150	bytes	Tbpe
Tbpe con bytes por firma digital.	170	bytes	Tbpefd
Tamaño en bytes de respuesta de estado servidor	275	bytes	Tbre
Tbre con bytes por firma digital.	295	bytes	Tbrefd
Tamaño en bytes de ACK estado de servidor	120	bytes	Tbackes
Tamaño en bytes de ACK respuesta de datos	120	bytes	Tbackrd

Ancho de banda total	316025	Bps	ABtotBps
	2528200	bps	ABtotbps=(Tbpefd+Tbrefd+((Tbpdfd+Tbrdfd)*(1000/Tm))+Tbackes+((Tbackrd*(1000/Tm)))*8
	2.5282	Mbps	
	2528.2	Kbps	

4 CAPÍTULO 4: SIMULACIÓN DE ALTERNATIVAS DE SISTEMAS SCADA CON OPC UA.

4.1 INTRODUCCIÓN A LA SIMULACIÓN DE ALTERNATIVAS DE SISTEMAS SCADA CON OPC UA.

En el presente capitulo se realizará una simulación entre un programa desarrollador de sistemas SCADA con un cliente OPC UA como driver de comunicaciones embebido y un controlador industrial basado en PC con un servidor OPC UA ejecutándose en máquinas virtuales de dos diferentes computadoras portátiles, con esta simulación se analiza la posibilidad de utilizar el protocolo OPC UA como remplazo del protocolo OPC tradicional actualmente utilizado.

Además, se diseña y crean páginas de interfaz hombre máquina para simular las diferentes capacidades de sistemas SCADA que tienen la opción de comunicarse con el protocolo OPC UA, como ejemplo de la simulación se toma el proceso de laminación en caliente para la fabricación de varillas de acero de construcción. Cada página posee datos de lectura como variables de monitoreo y datos de escritura como botones con sus respectivas variables que se enviarán hacia el controlador industrial basado en PC en el cual se ejecuta un servidor OPC UA.

También se realizará una captura de paquetes mediante el programa ¨Wireshark¨ para observar el manejo del ancho de banda utilizado entre el sistema SCADA y el controlador industrial.

4.2 DEFINICIÓN Y PARTES CONSTITUTIVAS DE UN SISTEMA SCADA.

SCADA son las iniciales de “Supervisory Control and Data Acquisition” y su principal función es la adquirir datos de dispositivos como sensores, válvulas y motores eléctricos para su posterior visualización por medio de una interfaz gráfica desarrollada

sobre un software especializado de interfaces hombre maquina (HMI). Además de la visualización de las variables de un proceso, también posee el control remoto como por ejemplo el prender y apagar de los motores eléctricos del proceso. (Schneider Electrics, 2012)

Un sistema SCADA posee 4 partes o componentes:

- Sensores en el proceso (Instrumentación)
- Controladores Lógicos Programables (PLCs) y/o Unidades Terminales Remotas (RTUs)
- Redes de comunicaciones y telemetría.
- Plataforma de software HMI.

En la siguiente figura 4.1 se puede observar un diagrama de las partes de un sistema SCADA.

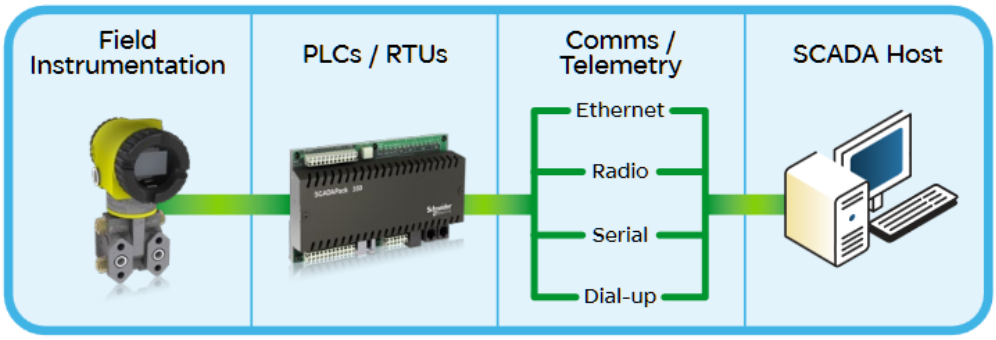

Figura 4.1: Componentes Principales de un sistema SCADA. Tomado de: SCADA Systems. Schneider Electrics, 2012.

En la siguiente simulación se utiliza un controlador industrial (PLC) que se ejecuta sobre una computadora con el sistema operativo Windows 2008 R2 por medio de un software "Runtime" llamado Twincat V3 de la empresa Bechkoff, el cual se comunicará sobre una red Ethernet hacia el software HMI llamado Genesis V64 de la empresa ICONICs.

4.3 USO Y CONFIGURACIÓN DE COMUNICACIÓN OPC UA EN EL CLIENTE OPC UA DEL SISTEMA SCADA SELECCIONADO.

4.3.1 GENESIS64 de ICONICS

Genesis64 es el conjunto de aplicaciones para implementar aplicaciones SCADA desarrollado por la empresa ICONICS para sistemas operativos Windows ejecutados a 64 bits. Los módulos principales disponibles en Genesis64 son: GraphWorx64, TrendWorx64 y AlarmWorx64. GraphWorx64 es la aplicación utilizada para implementar las interfaces de usuario y las pantallas de ejecución del sistema hombre – máquina. La aplicación TrendWorx64 es la aplicación para implementar gráficos de tendencias de variables de proceso. Por último, AlarmWorx64 es la aplicación para la configuración de alarmas de proceso que serán visualizadas en la interfaz de usuario (Iconics HMI/SCADA Software Developer, 2012).

Para la conexión de datos hacia los controladores industriales, la aplicación GraphWorx64 posee un explorador de fuentes de datos en la cual se tiene incorporado un cliente OPC UA con el cual se puede conectarse con un servidor OPC UA. El explorador de fuentes de datos OPC UA se muestra en la figura a continuación.

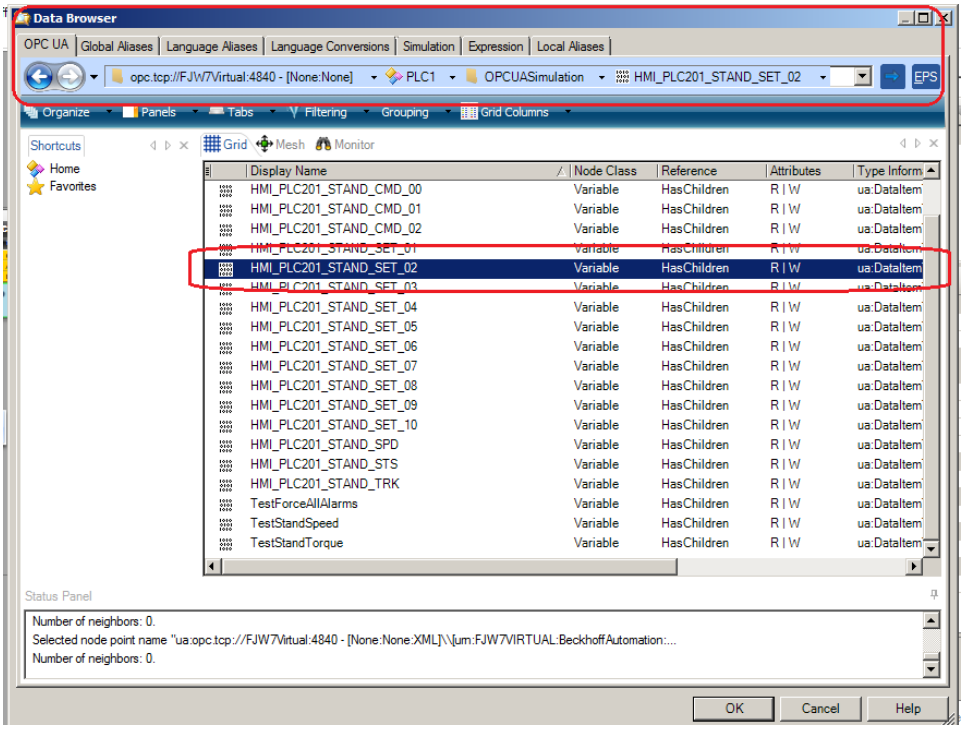

Figura 4.2: Explorador de datos OPC UA de la aplicación GraphWorx64 de la empresa ICONICS. Genesis64. Fuente autor.

4.4 DISEÑO DE LA INTERFAZ Y PROCESAMIENTO DE DATOS A UTILIZAR PARA SIMULACIÓN EN EL CLIENTE OPC UA DEL SISTEMA SCADA SELECCIONADO.

4.4.1 Página HMI de estados de funcionamiento de máquinas.

Para realizar la simulación se desarrolla una interfaz de usuario en la aplicación GraphWorx64 ejecutándose en una máquina virtual instalada con el sistema operativo Windows Server 2008 R2. La pantalla implementada es para el manejo de máquinas de laminación en caliente de hierro para la fabricación de varillas de construcción. Se realiza la pantalla de manejo de estas máquinas en las cuales se puede ver el estado de funcionamiento de cada motor y la velocidad a la cual esta cada una, además, se puede observar el funcionamiento si la maquina se encuentra en manual o automático.

La laminación en caliente de hierro es el proceso en el cual se pasa un lingote caliente de hierro, con aproximadamente 1200 grados centígrados, y se lo hace pasar por cajas de laminación. Cada caja de laminación posee dos cilindros, uno debajo de otro, con canales que permiten pasar al lingote de hierro caliente pero con otro tipo de forma y dimensiones reducidas. En la siguiente figura se muestra el proceso de laminación con la barra de hierro pasando por 3 cajas de laminación.

Figura 4.3: Proceso industrial de laminación en caliente de hierro. Tomado de: “Speed cascade control system for bar and wire rod mills”, ABB, 2012.

La descripción de la red utilizada para la simulación entre el sistema SCADA y el controlador lógico programable en PC se muestra en la figura siguiente.

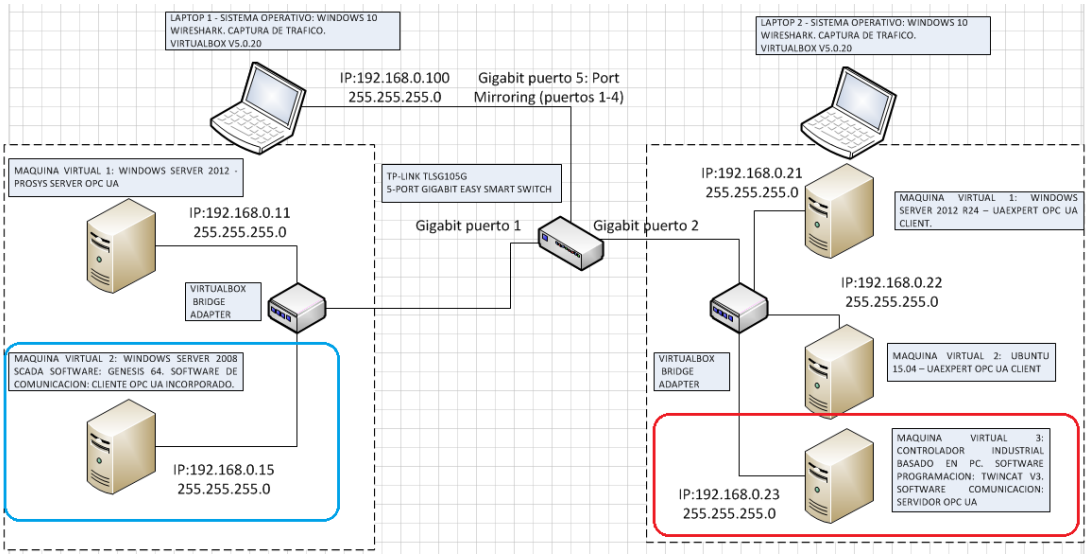

Figura 4.4: Esquema de la red LAN para la simulación de sistemas SCADA. Fuente el autor

La simulación está conformada por 10 cajas de laminación con su respectivo símbolo y la velocidad del motor conectada a la máquina. En la parte superior de la pantalla implementada se encuentra un resumen de las alarmas más recientes, debajo del resumen de alarmas se tiene una cinta de pestañas para cambiar entre páginas. En la página se tiene 4 pestañas: la pestaña del sinóptico, el gráfico de tendencias, una pantalla de una unidad de lubricación para las máquinas y un historial de alarmas. En la siguiente figura se muestra la pantalla desarrollada.

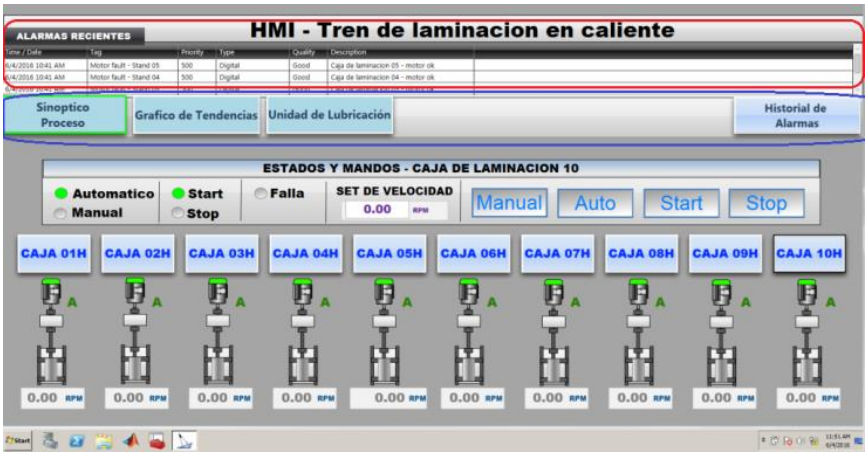

Figura 4.5: Página HMI de funcionamiento de máquinas implementado en la aplicación GraphWorx64 de la empresa ICONICS. Genesis64. Fuente autor.

En cada animación de la página de estados de funcionamiento se debe especificar la fuente de datos que en este caso es una variable del servidor OPC UA corriendo sobre un controlador industrial basado en PC corriendo el runtime del software Twincat. En la siguiente figura se muestra la especificación del ítem OPC UA.

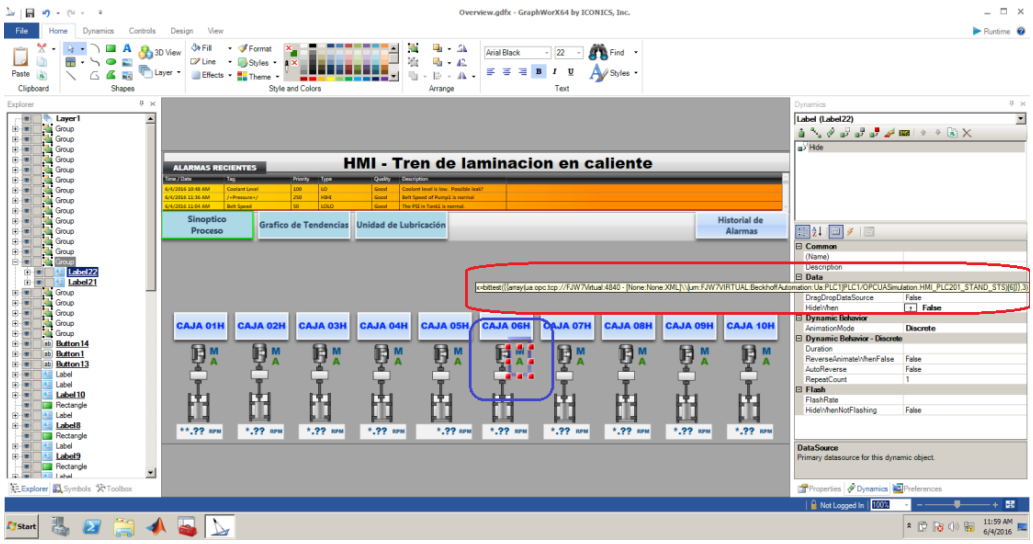

Figura 4.6: Especificación como fuente de datos de un ítem OPC UA para una animación en la página HMI de estados. Genesis64. Fuente autor.

4.4.2 Página HMI de alarmas de funcionamiento de máquinas.

En la siguiente figura se muestra el listado de las alarmas de funcionamiento que se podrá visualizar en el sistema SCADA.

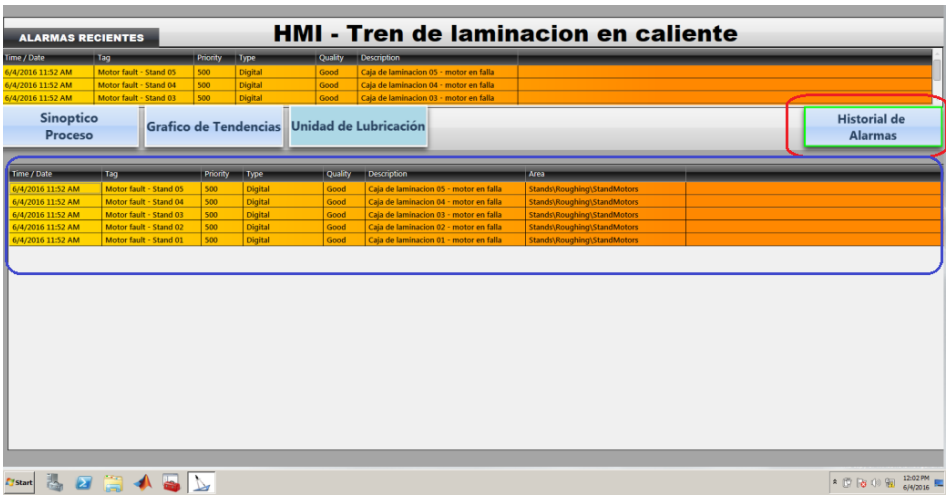

Figura 4.7: Página HMI de historial de alarmas implementado en la aplicación AlarmWorx64 de la empresa ICONICS. Genesis64. Fuente autor.

4.4.3 Página HMI de gráficos de tendencias de variables de máquinas.

En la siguiente figura se muestra la página HMI del grafico de tendencias de la velocidad de la caja de laminación 01.

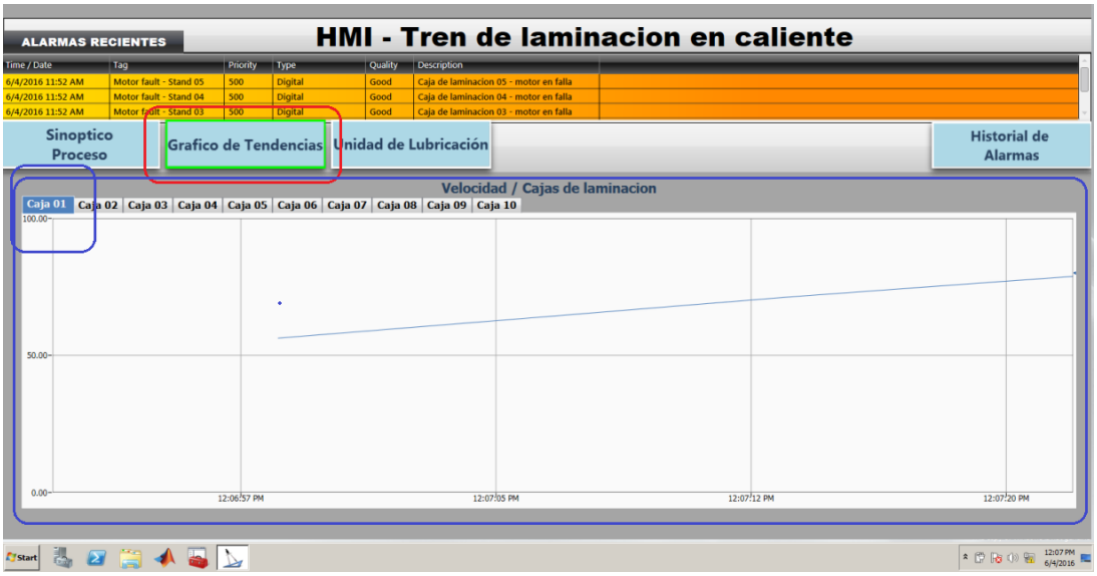

Figura 4.8: Página HMI del gráfico de tendencias implementado en la aplicación TrendWorx64 de la empresa ICONICS. Genesis64. Fuente autor.

4.4.4 Página HMI de mandos para accionamientos en máquinas.

En la siguiente figura se muestra la página HMI para los mandos hacia las maquinas del proceso conocidas como cajas de laminación. Para cada caja existen botones que envían mandos al servidor OPC UA, estos botones son los de los comandos de Automático, Manual, Start y Stop. Además de los botones, se muestran los estados de la máquina y el campo de valor deseado de la velocidad de la caja de laminación.

Figura 4.9: Página HMI de comandos hacia las máquinas de proceso implementado en la aplicación GraphWorx64 de la empresa ICONICS. Genesis64. Fuente autor.

En la siguiente figura se muestra la configuración del dato enviado hacia el servidor OPC UA el momento de que se ejecute el evento de presionar sobre uno de los botones de la página HMI de comandos de máquinas.

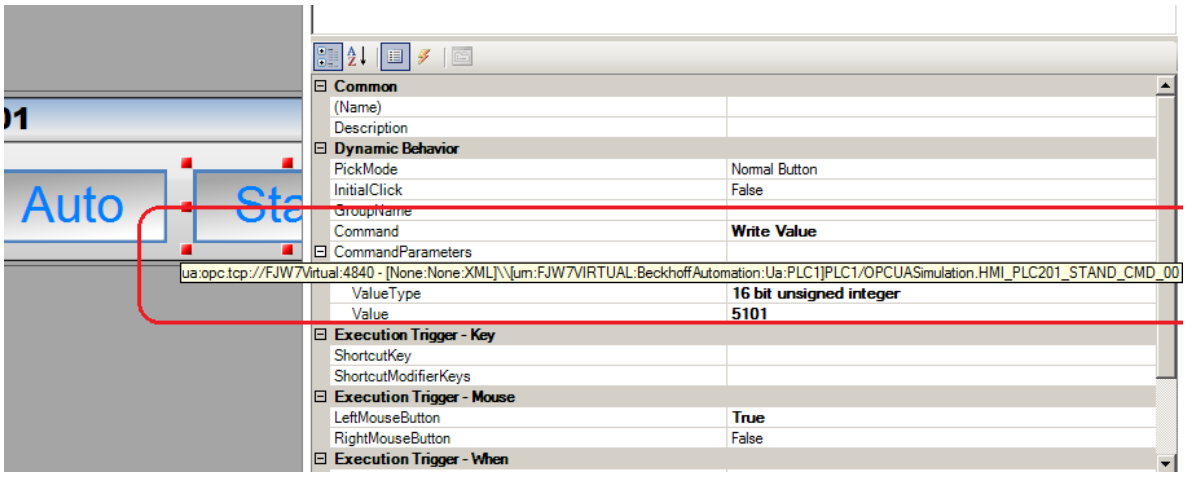

Figura 4.10: Configuración de botones de la página HMI de comandos hacia las máquinas de proceso implementado en la aplicación GraphWorx64 de la empresa ICONICS. Genesis64. Fuente autor.

4.5 USO Y CONFIGURACIÓN DEL CONTROLADOR INDUSTRIAL BASADO EN PC COMO SERVIDOR OPC UA SELECCIONADO.

4.5.1 SoftPLC Runtime Twincat V3.

Twincat V3 es el software desarrollado por la empresa alemana Beckhoff para la configuración de los controladores industriales de la misma empresa, y que además, posee un software runtime para que se ejecute sobre un sistema operativo y tenga las capacidades de un controlador industrial basado en PC.

Para la simulación, se instala el Runtime del software Twincat en una máquina virtual y se programa para ejecutar la recepción y envío de datos por medio del protocolo OPC UA. Para configurar una variable definida en el programa del controlador industrial, esta debe tener un atributo antes de la declaración de la variable, en la siguiente figura se muestra una de las variables utilizadas en el programa del controlador con el atributo

(attribute ‘OPC.UA.DA’ := ‘1’) el cual define que esa variable va a comunicarse por el protocolo OPC UA y puede ser leída y escrita desde un cliente OPC UA (Bechkoff, 2015).

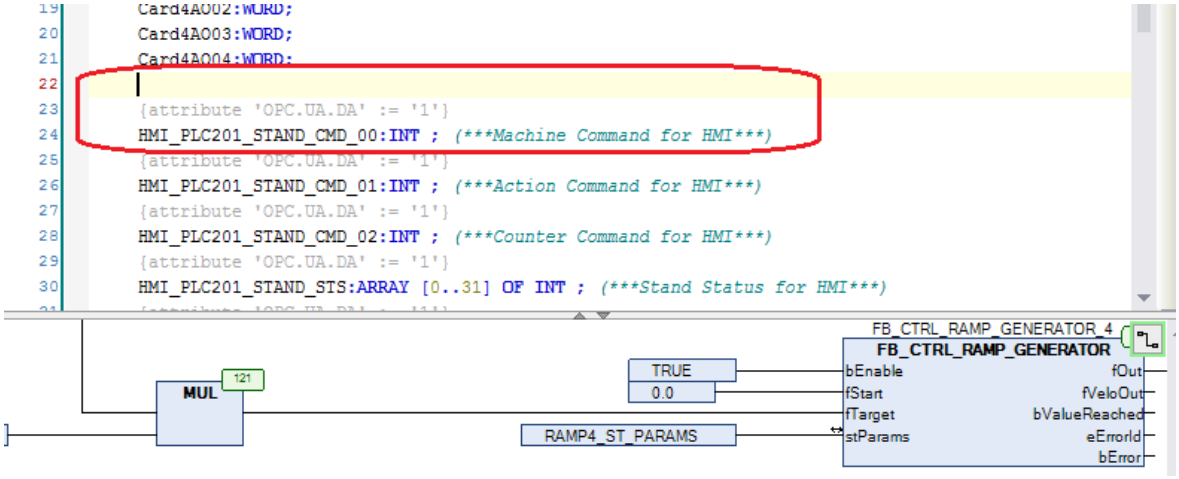

Figura 4.11: Configuración de variables OPC UA en el controlador industrial implementado en el software Twincat V3 de la empresa Bechkoff. Twincat V3. Fuente autor.

4.5.2 Driver de comunicación OPC UA para Runtime Twincat V3

EL software runtime Twincat V3 de la empresa Bechkoff posee las funcionalidades de un controlador industrial basado en PC, en el caso de la comunicación mediante el protocolo OPC UA necesita de una funcionalidad extra de comunicación para lo cual se le debe instalar el driver de comunicación OPC UA.

En la siguiente figura se muestra la instalación del driver de comunicación y el funcionamiento del driver de comunicación.

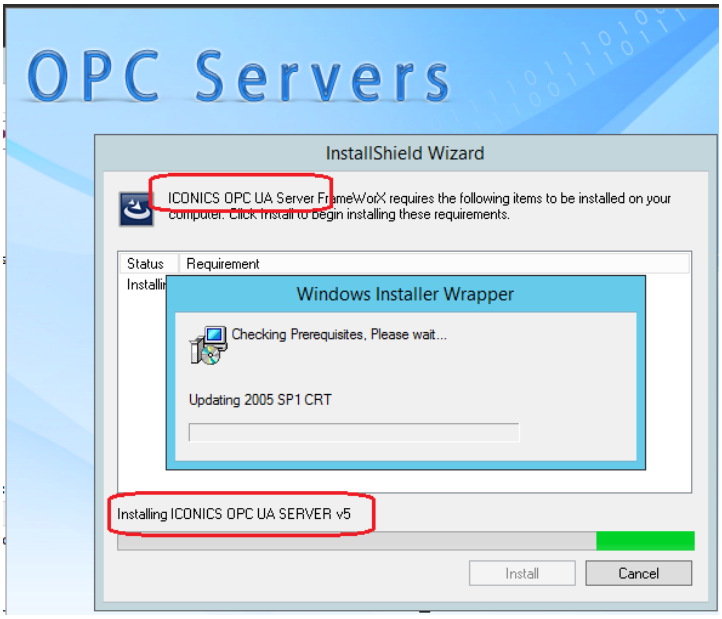

Figura 4.12: Instalación del driver de comunicación OPC UA en el controlador industrial implementado en el software Twincat V3 de la empresa Bechkoff. Twincat V3. Fuente autor.

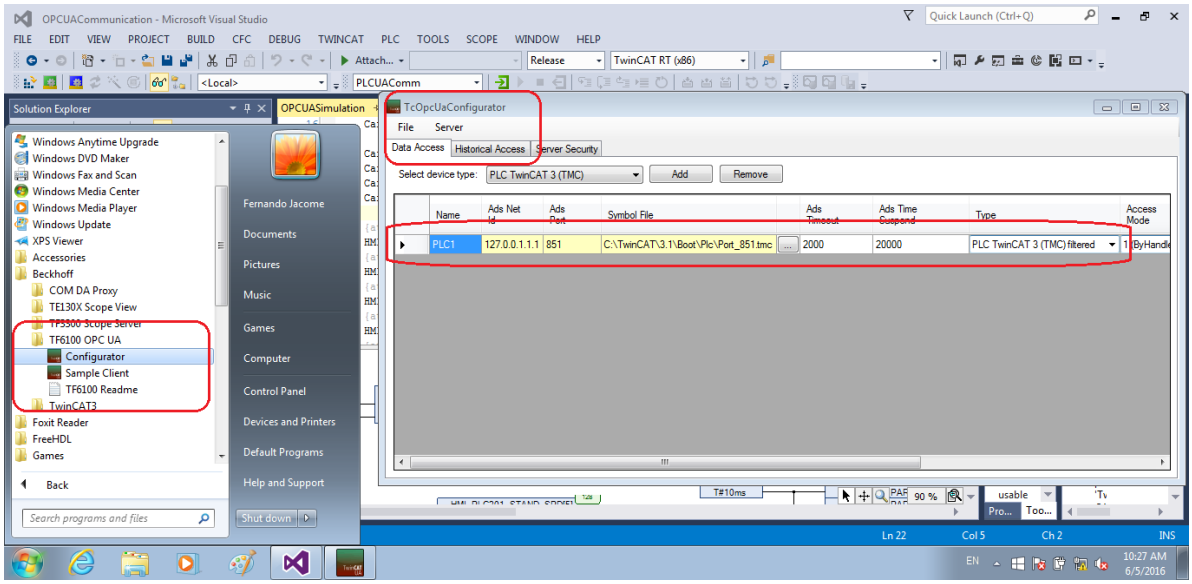

Figura 4.13: Consola del driver de comunicación OPC UA en el controlador industrial implementado en el software Twincat V3 de la empresa Bechkoff. Twincat V3. Fuente autor.

4.6 USO Y CONFIGURACIÓN DE COMUNICACIÓN OPC UA EN EL CONTROLADOR INDUSTRIAL BASADO EN PC.

4.6.1 Programación, recepción y envíos de datos OPC UA con Twincat V3.

Para la comunicación OPC UA en Twincat V3, el driver de comunicación que implementa un servidor OPC UA utiliza el puerto 4840 para el establecimiento de la conexión y como métodos de autenticación permite el acceso anónimo, y con usuario y contraseña. En cuanto a las políticas de seguridad posee las opciones de: sin seguridad, firmado - encriptado Basic128Rsa15, y firmado - encriptado Basic256.

De igual forma desde la consola del driver de comunicación OPC UA se puede configurar el usuario y contraseña en el modo de autenticación y también se tiene el manejo de los certificados confiables a conectarse al servidor OPC UA del controlador industrial como se puede observar en la figura siguiente.

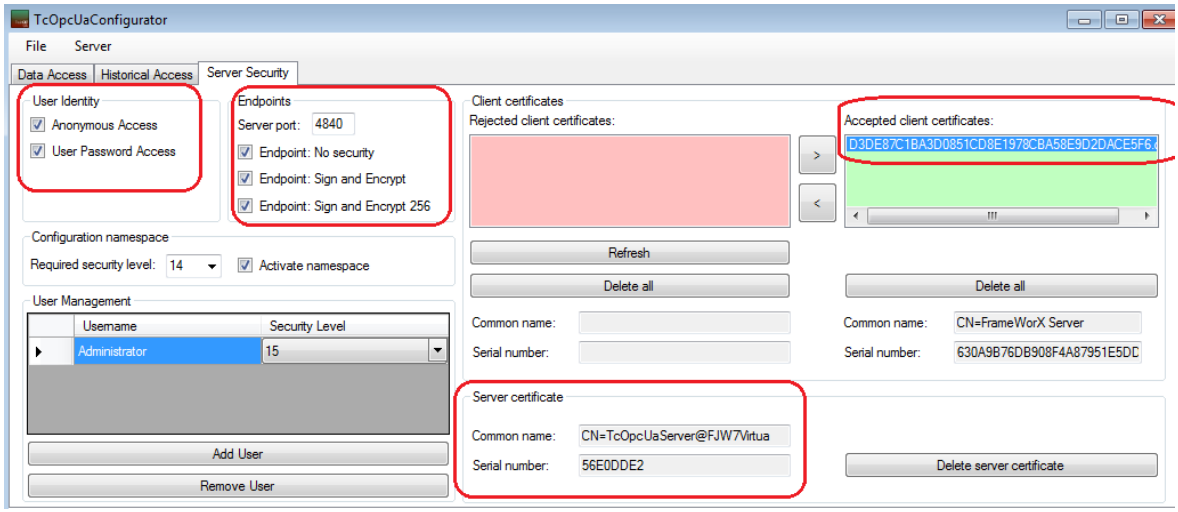

Figura 4.14: Configuración de usuarios y certificados en el driver de comunicación OPC UA del controlador industrial implementado en el software Twincat V3 de la empresa Bechkoff.

Twincat V3. Fuente autor.

En la siguiente figura se muestra el explorador OPC UA del software SCADA Genesis64 para el acceso a las variables del controlador industrial ejecutándose Twincat V3.

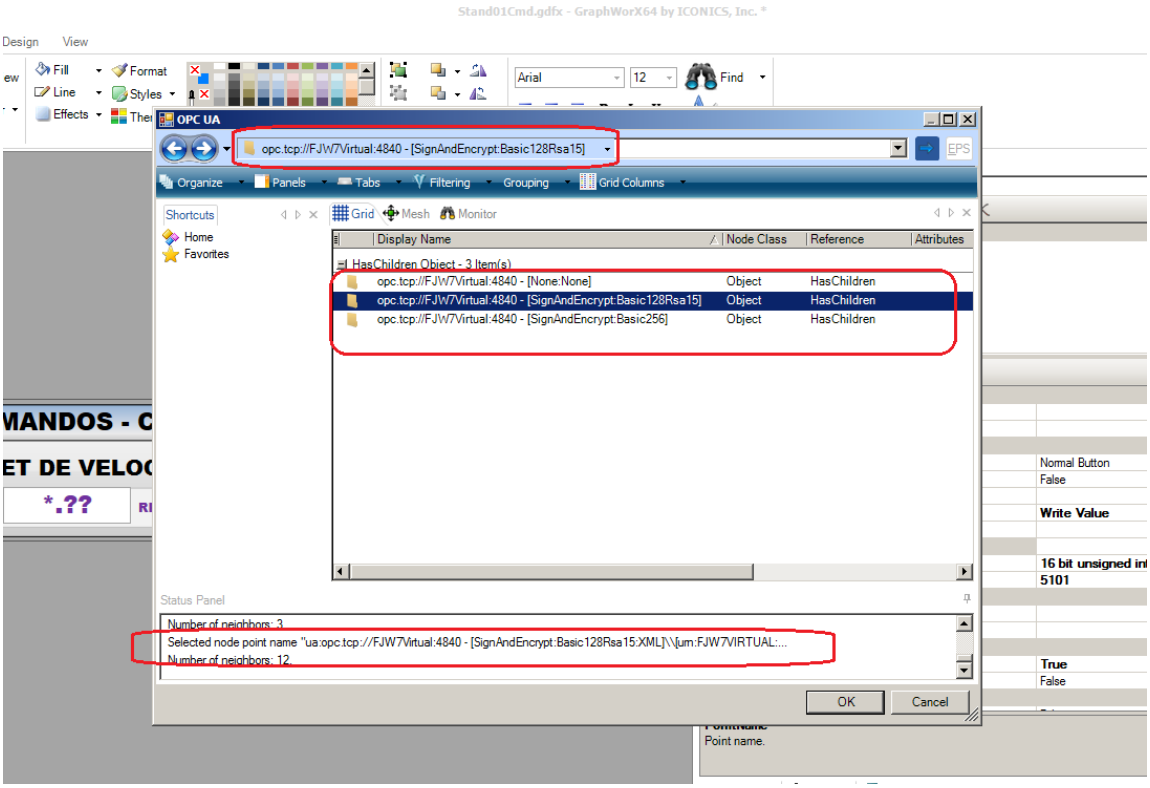

Figura 4.15: Explorador cliente de datos OPC UA del software SCADA Genesis64. Genesis64. Fuente autor.

Para la recepción de los datos enviados desde el sistema SCADA se programaron las variables HMI_PLC201_STAND_CMD del tipo de datos INT (Entero de 16 bits). Para el manejo de estados y alarmas de las maquinas simuladas hacia el sistema SCADA se crearon las variables HMI_PLC201_STAND_STS y HMI_PLC201_STAND_ALM. En el tratamiento del dato de la velocidad de cada máquina simulada hacia el sistema SCADA se crearon las variables HMI_PLC201_STAND_SPD. Por último para el envío de las velocidades deseadas de los motores eléctricos, desde el sistema SCADA hacia el programa de las maquinas simuladas, se crearon las variables HMI_PLC201_STAND_SET en el controlador basado en PC. En la siguiente figura se muestra la definición de las variables anteriormente mencionadas y la aplicación del atributo OPC UA para su comunicación.

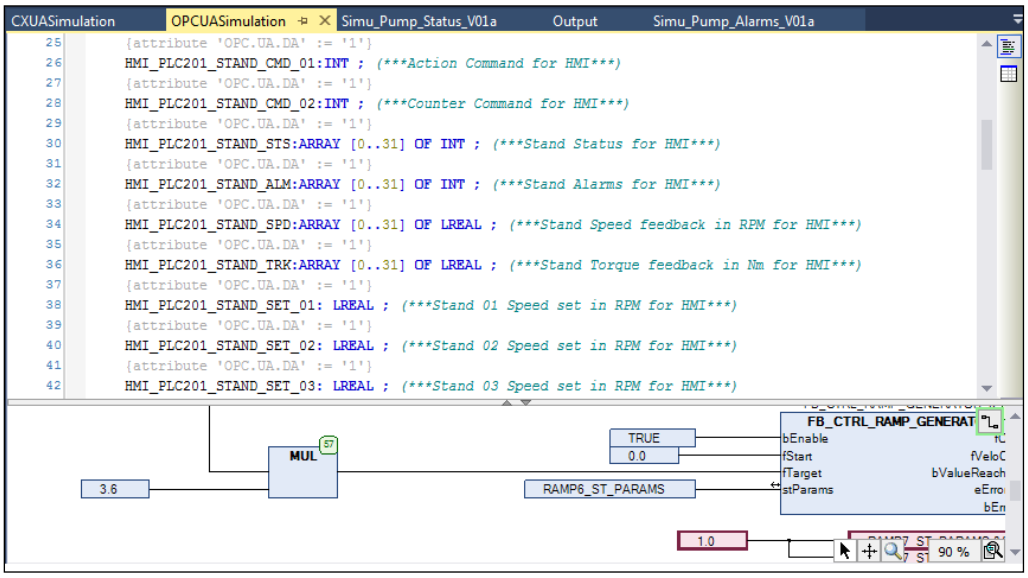

Figura 4.16: Declaración de variables OPC UA en el controlador industrial basado en PC.

Twincat V3. Fuente autor.

4.7 DESCRIPCIÓN DEL ESCENARIO DE PRUEBA DE SIMULACIÓN SCADA.

Las pruebas se realizaron con el sistema SCADA ejecutando el software Genesis64 instalado sobre una máquina virtual configurada con la dirección ip: 192.168.0.15 con mascara de subred 255.255.255.0 y estableciendo la comunicación OPC UA con el controlador industrial basado en PC ejecutando Twincat V3 instalado sobre otra máquina virtual con dirección ip: 192.168.0.23, se tomaron capturas de paquetes de la lectura de las variables en una de las pantallas del sistema SCADA por medio del software Wireshark. Las pruebas se realizan con ningún modo de seguridad, es decir los datos no están firmados ni encriptados para observar el comportamiento en el consumo de ancho de banda. En la figura 4.17 se muestra un diagrama del escenario de prueba de la simulación SCADA.

Al abrir la conexión entre servidor y cliente OPC UA con ningún modo de seguridad y ninguna política de seguridad, el servidor y cliente no intercambian certificados de instancia de aplicación que valida la autenticidad de las aplicaciones que están incluidas en la comunicación OPC UA. Si se realiza con un modo de seguridad firmado-cifrado, se debe intercambiar los certificados de aplicación del cliente con el servidor y viceversa, para esto se puede mover manualmente el certificado de la aplicación del cliente y copiarlo en la

máquina que se esté ejecutando la aplicación del servidor OPC. Tanto el servidor como el cliente poseen carpetas de archivos para la funcionalidad de la administración de los certificados. Para que los certificados sean considerados confiables, estos deben ser intercambiados y copiados en la carpeta de manejo de certificados y dentro de la carpeta de certificados confiables (Trust).

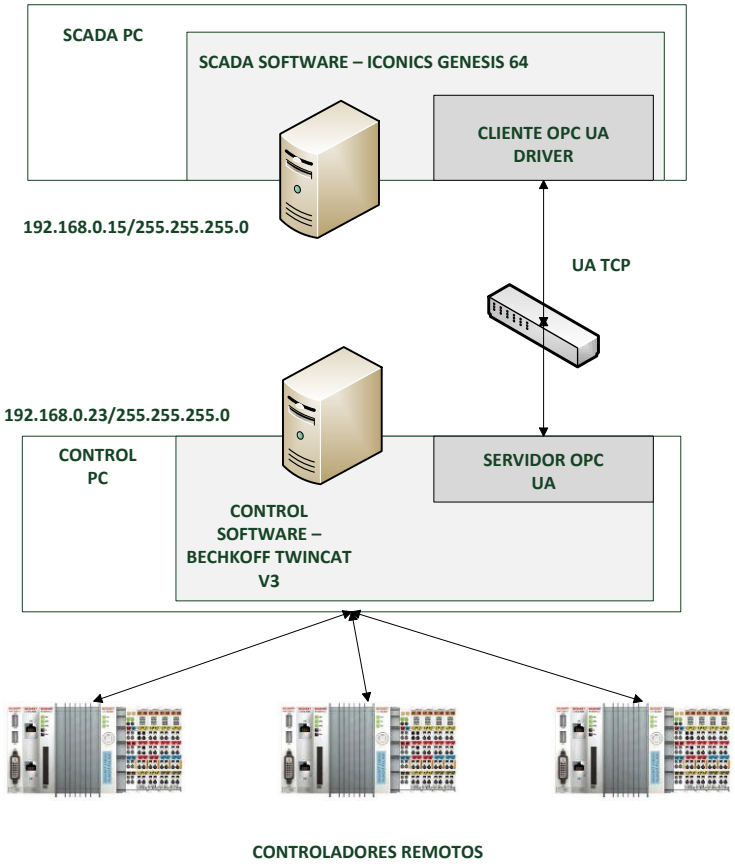

Figura 4.17: Diagrama del escenario de prueba para la simulación SCADA. Fuente autor

4.7.1 Escenario de prueba 18.

4.7.1.1 *Escenario de prueba 18a (Lectura de un arreglo de 32 datos).*

En esta prueba se realiza el análisis de **la comunicación OPC UA del sistema SCADA con el controlador industrial basado en PC sin modo o política de seguridad, usuario anónimo y tiempo de muestreo de 100 milisegundos para un arreglo de 32 datos del tipo doble**.

Al realizar la simulación se observó que la publicación de la petición y respuesta a la lectura de los datos de la página del sinóptico del sistema SCADA empieza a enviar y recibir datos el momento que los datos cambian, si los datos no cambian el envío y recepción de datos es casi mínima solo para interrogar el estado del servidor y del estado de la publicación de los datos de lectura. Al no configurar un tiempo de muestreo, el cliente OPC UA empieza a utilizar el tiempo de muestreo de 100 milisegundos, obteniendo hasta 10 publicaciones de los datos de lectura desde el servidor OPC UA del controlador industrial hacia el cliente OPC UA del sistema SCADA. La captura de los datos así como el gráfico del ancho de banda utilizado se pueden observar en las siguientes figuras 4.18 y 4.19.

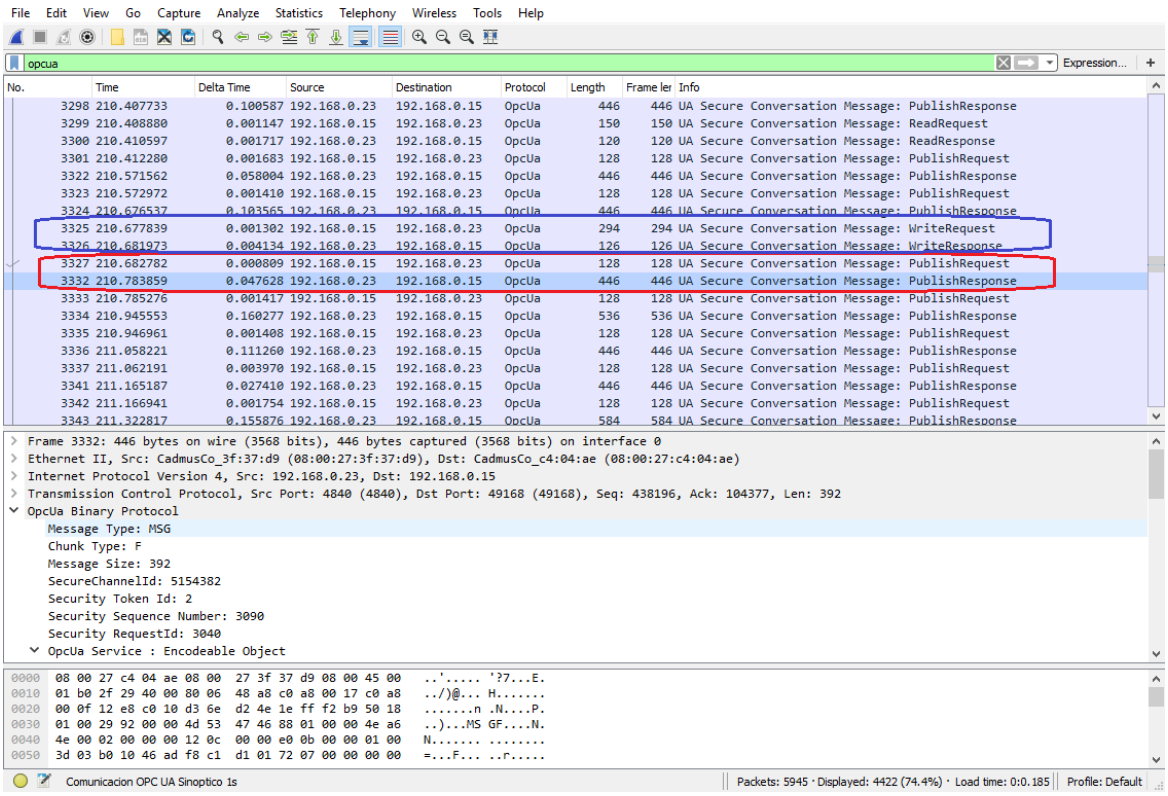

Figura 4.18: Captura el tráfico OPC UA. Escenario de prueba 18a. Fuente autor.

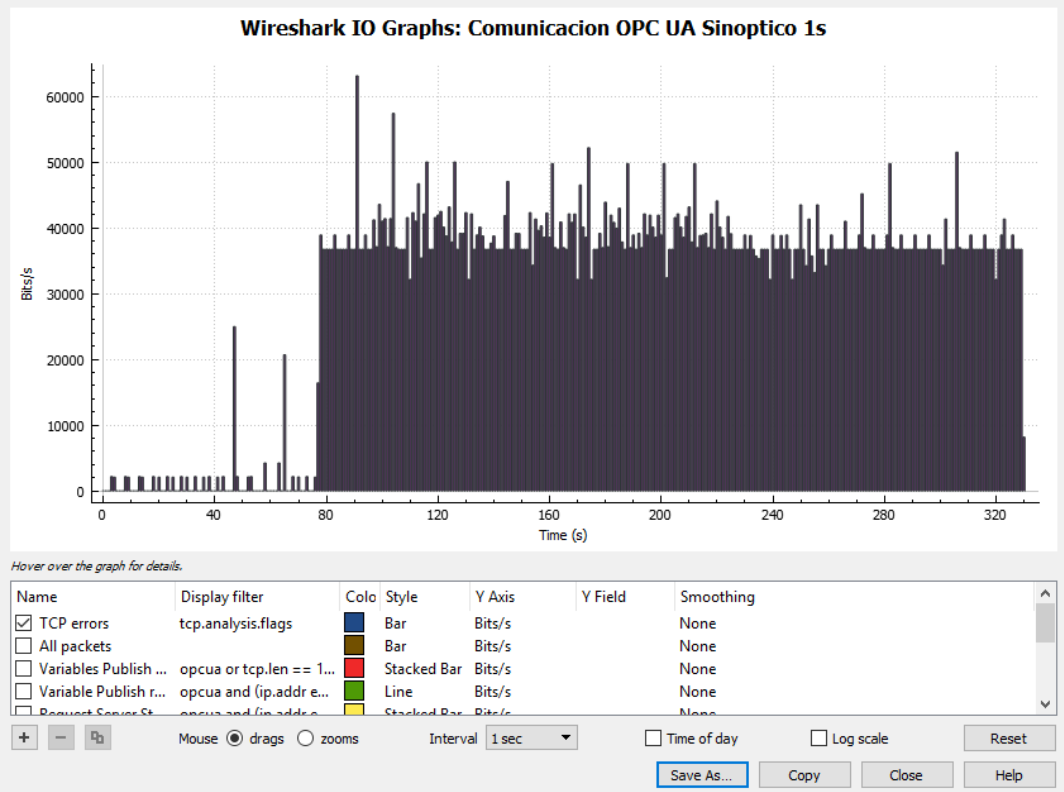

Figura 4.19: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 18a. Fuente autor.

Una diferencia con respecto a la simulación de datos de tipo doble independientes realizadas en los escenarios de pruebas anteriores de simulación de alternativas OPC UA, (los cuales usaban 30 bytes por cada dato tipo doble), con respecto a la simulación de sistemas SCADA, es la utilización de arreglos de datos de tipo doble que se utilizó en las variables del controlador industrial. Al observar la captura de datos se pudo constatar que al utilizar arreglos de datos tipo doble solo se utiliza 8 bytes por cada dato de tipo doble con respecto a usar datos de tipo doble independientes con lo cual se disminuyó la longitud en bytes del mensaje de respuesta a la publicación de lectura de los datos.

En el caso del sistema SCADA el monitoreo de las velocidades de las maquinas simuladas se configuro como un arreglo de 32 datos de tipo doble, al utilizar 8 bytes por cada dato de tipo doble por la respuesta OPC UA se obtiene 32 * 8 = 256 bytes de datos más 14 bytes para definir el dato y diagnóstico de estado del arreglo se obtienen 270 bytes adicionalmente se tienen los 122 bytes de las cabeceras OPC UA con lo cual se obtienen 392 bytes y por último se adicionan 54 bytes de las demás cabeceras TCP, IP y Ethernet

dando un total de 446 bytes para la respuesta de la publicación de lectura de datos. Al dato anterior se debe adicionar los 128 bytes de la petición de publicación de los datos con lo cual resultan un total de 574 bytes o 574 x 8 = 4592 bits por cada mensaje y teniendo en cuenta que se envía un mensaje cada 100 milisegundos se tienen 10 mensajes por segundo con un total de 45920 bps.

Además de la lectura de datos se tienen botones que envían un dato hacia el controlador industrial, esto únicamente se activa el momento de presionar un botón sobre la pantalla de comandos. Para enviar estos datos de mandos se utiliza una solicitud y respuesta de escritura de datos OPC UA. El tiempo que se demora entre la petición y la respuesta de esta escritura de datos es de aproximadamente 10 milisegundos.

4.7.2 Escenario de prueba 19.

4.7.2.1 *Escenario de prueba 19a (Lectura de un arreglo de 32 datos).*

En este escenario de prueba se realiza la conexión **con mensaje firmado y cifrado, política de seguridad Basic128RSA15, usuario anónimo y tiempo de muestreo de 100 milisegundos**. En el escenario de prueba 19 se utiliza firmado y cifrado digital en la comunicación de datos de la pantalla de Sinóptico del sistema SCADA, por lo cual en cada petición y respuesta del valor de las variables se debe añadir 24 bytes extras de la firma digital. Al tener los 446 bytes del escenario de prueba anterior (Sin firmado digital) se debe añadir los bytes de la firma digital del escenario de prueba actual dándonos un total de 470 bytes. Además del firmado digital se cifro a los datos, por lo cual en la captura de datos se observa el contenido cifrado de los paquetes.

Con 150 bytes de la petición de los datos y 470 bytes de la respuesta y publicación se tiene un total de 620 bytes o 620 x 8 = 4960 bits por cada mensaje y teniendo en cuenta que se envía un mensaje cada 100 milisegundos se tienen 10 mensajes por segundo con un total de 49600 bps.

En la figura 4.20 y 4.21 se pueden observar la captura de los datos y el ancho de banda utilizado.

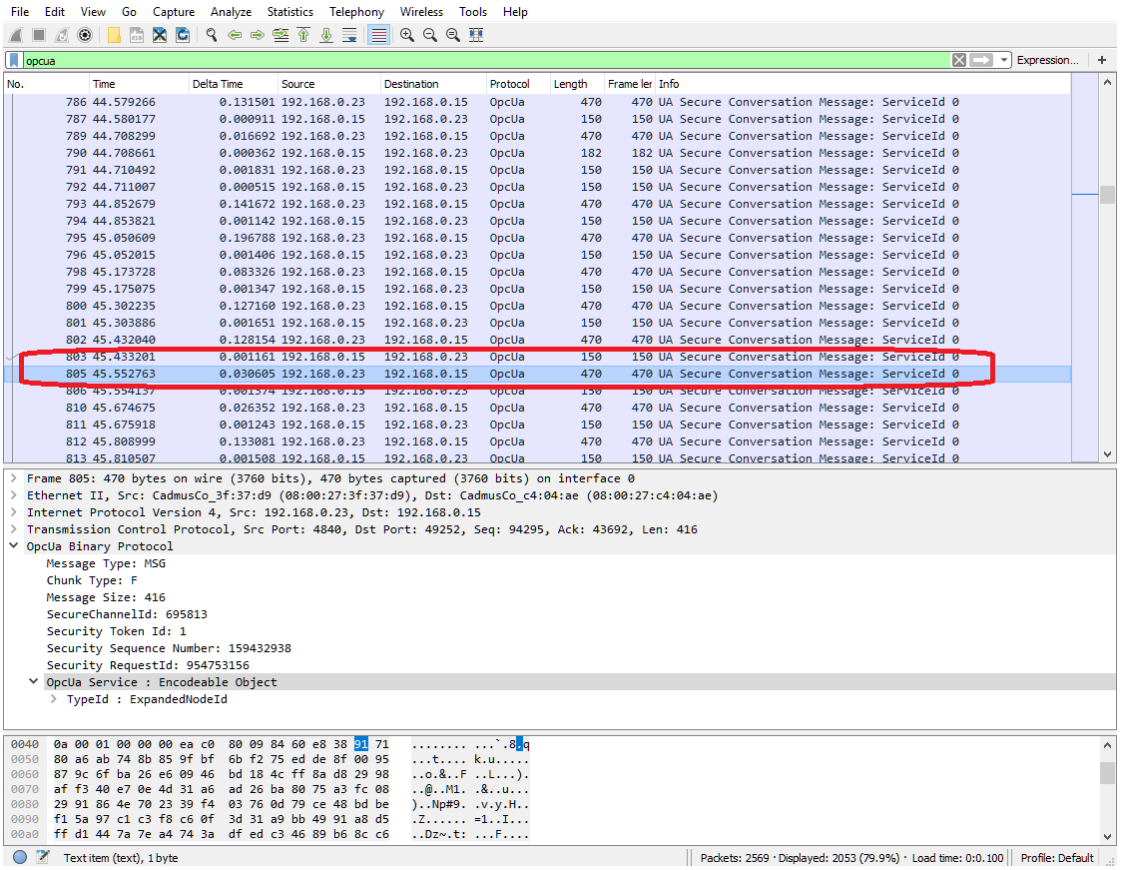

Figura 4.20: Captura el tráfico OPC UA. Escenario de prueba 19a. Fuente autor.

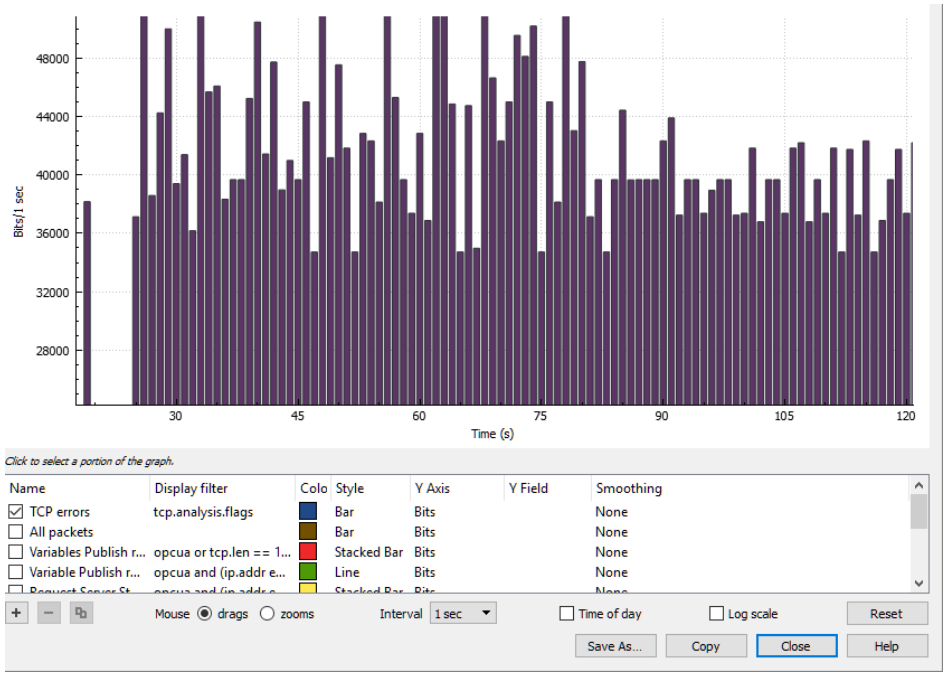

Figura 4.21: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 19a. Fuente autor.

4.7.3 Escenario de prueba 20.

4.7.3.1 *Escenario de prueba 20a (Lectura de un arreglo de 32 datos).*

En este escenario de prueba se realiza la conexión **con mensaje firmado y cifrado, política de seguridad Basic256, usuario anónimo y tiempo de muestreo de 100 milisegundos**. Al cambiar la política de seguridad de Basic128Rsa15 a Basic256 se fortalece el cifrado de los datos a comunicar. Al observar la captura de datos con la nueva política de seguridad se puede notar que el tamaño del paquete es igual al anterior escenario de prueba 19, por lo tanto la firma digital utiliza el mismo número de bytes y el ancho de banda utilizado es el mismo. Al observar el tiempo de ida y retorno (Round Trip time) se puede observar un valor máximo de 21 ms para el caso de comunicación con política de seguridad Basic128Rsa15 y un máximo de 220 ms para el caso con política de seguridad Basic256. En las siguientes figuras 4.22, 4.23 y 4.24 se puede observar la captura de datos, el ancho de banda utilizado y la comparación de tiempo de ida y retorno para las dos políticas de seguridad mencionadas anteriormente.

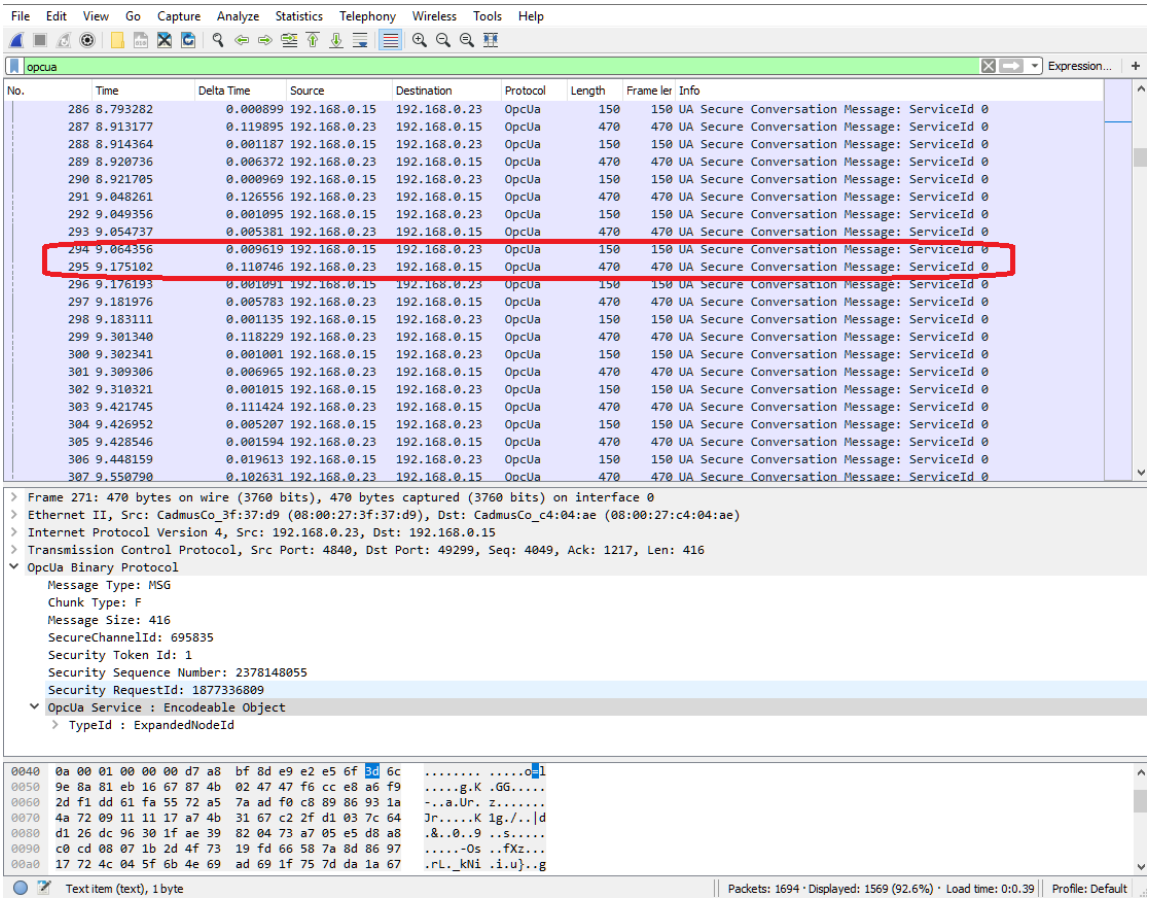

Figura 4.22: Captura el tráfico OPC UA. Escenario de prueba 20a. Fuente autor.

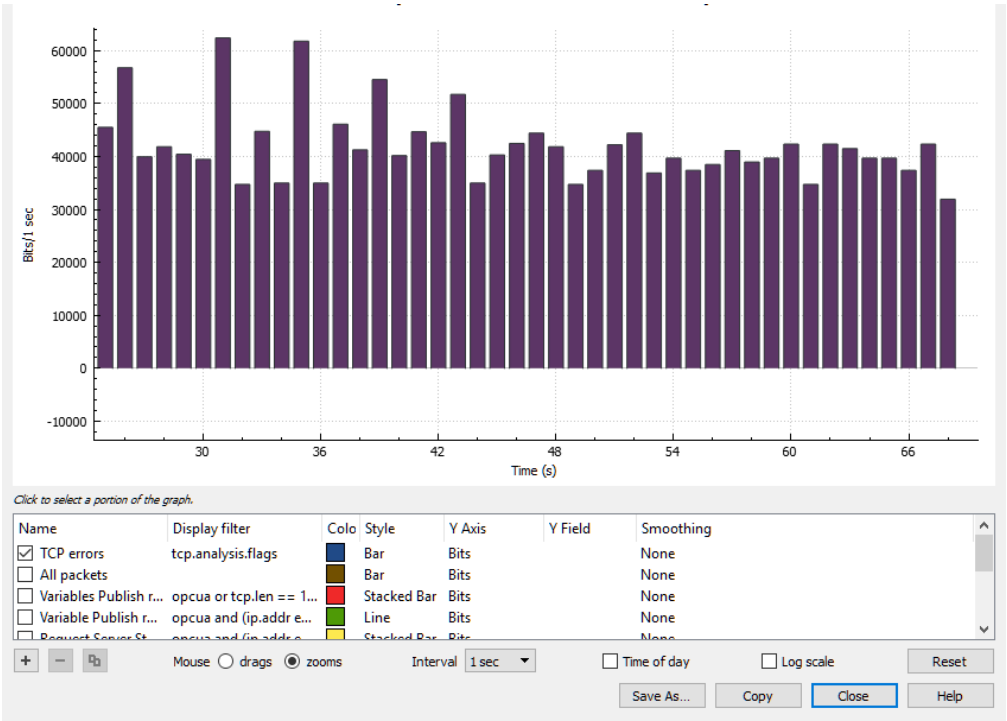

Figura 4.23: Uso del ancho de banda del tráfico OPC UA. Escenario de prueba 20a. Fuente autor.

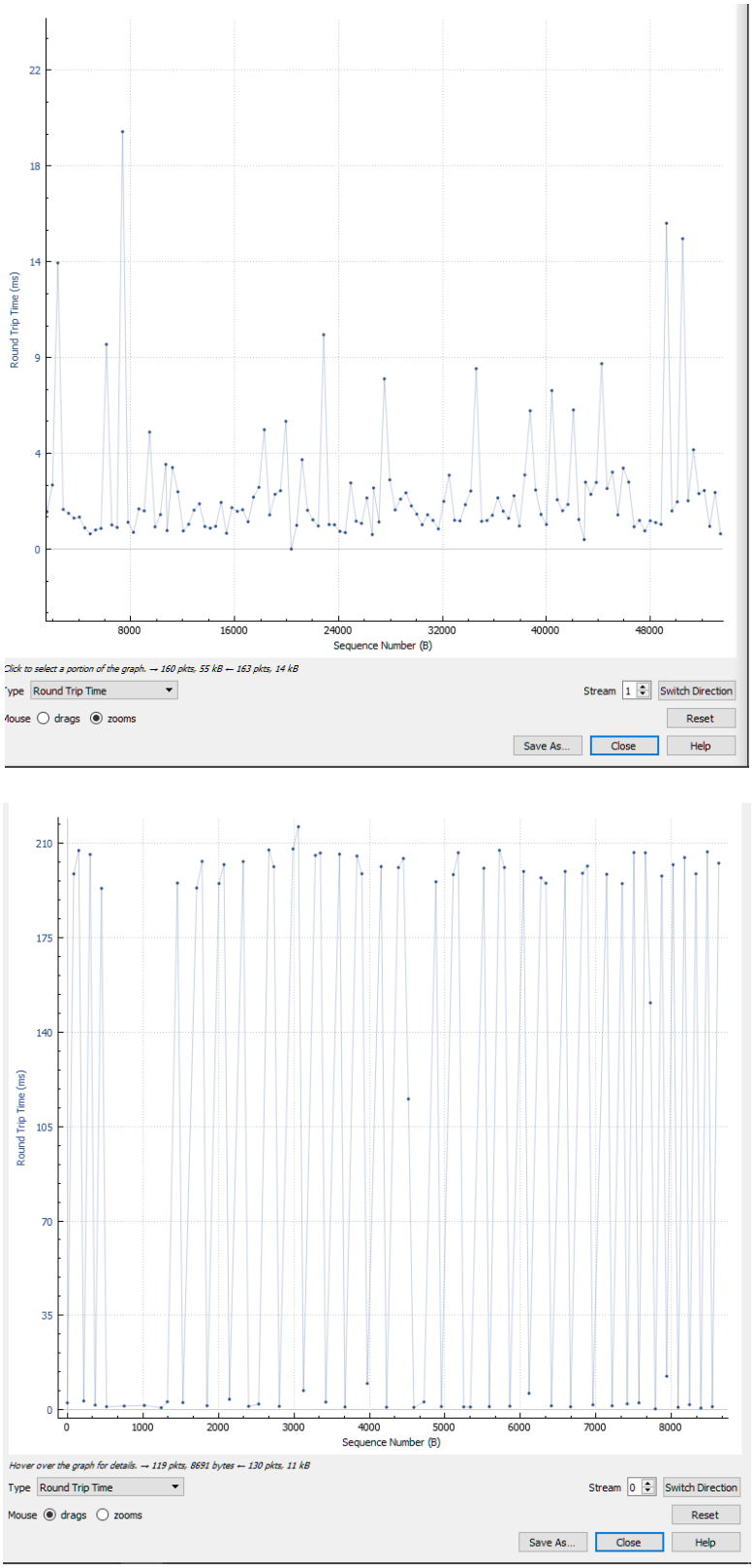

Figura 4.24: Gráfico del tiempo de ida y retorno para el caso de una comunicación de datos SCADA firmados-cifrados y política de seguridad Basic128Rsa15 (arriba) vs política de seguridad Basic256 (abajo). Escenario de prueba 20. Fuente autor.

4.7.4 Análisis comparativo de los escenarios de prueba con lectura de datos con firmado digital y cifrado con simulación SCADA.

En la siguiente Tabla 4.1 se pueden observar los escenarios de pruebas en los cuales se tiene firma digital y cifrado en los datos. Una de las diferencias notables se observan en el tiempo de ida y retorno de los mensajes de los datos en donde se puede observar un aumento significativo probablemente por el procesamiento de cifrado y descifrado y el manejo de arreglo de datos.

Tabla 4.1: Tabla comparativa de los escenarios de prueba con lectura de datos con firmado digital y cifrado de la simulación SCADA. Fuente el autor.

Escenario de prueba	Modo de seguridad	Política de seguridad	Autenticación de usuario	Tiempo de muestreo	Numero de datos	Numero de paquetes	Número de peticiones por segundo	Numero de bytes por datos	Ancho de banda (Petición lectura de datos)	Ancho de banda (Respuesta lectura de datos)	Ancho de banda (Petición estado del servidor)	Ancho de banda (Respuesta estado del servidor)	Ancho de banda total (Petición datos y estado servidor)	Ancho de banda total (Respuesta datos y estado servidor)	Observaciones
				(ms)	(Arreglo Tipo doble)				(bps)	(bps)	(bps)	(bps)	(bps)	(bps)	
Prueba 18	Mensaje sin Firmado ni Cifrado	Ninguno	Anónimo	100	32	1	10	256	10240	35680	0	0	10240	35680	Aumento del tiempo de ida y retorno de los datos debido al proceso de cifrado y manejo de arreglos
Prueba 19	Mensaje Firmado y Cifrado	Basic128RSA15	Anónimo	100	32	1	10	256	12000	37600	0	0	12000	37600	
Prueba 20	Mensaje Firmado y Cifrado	Basic256	Anónimo	100	32	1	10	256	12000	37600	0	0	12000	37600	

5 CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES.

5.1 CONCLUSIONES

- La primera concusión de este trabajo está relacionada con la facilidad de uso que presenta OPC UA. Debido a que las implementaciones de este protocolo han sido desarrolladas en sistemas abiertos como Linux y el lenguaje java. El esfuerzo y la complejidad de la configuración se reducen substancialmente, por ejemplo, no se necesita ninguna configuración de permisos especiales del sistema operativo, toda la configuración de seguridad se la realiza en la aplicación del servidor y del cliente OPC UA, lo cual reduce el esfuerzo de la solución y la hace más mantenible.

- Se pudo observar que la comunicación OPC UA es amigable aun con la presencia de cortafuegos de red y la configuración de permiso de acceso es por medio de puertos definidos y conocidos, no se generan puertos aleatorios que darían una complejidad al momento de permitir el acceso por medio de esos puertos. Para la conexión por medio NAT se debe configurar la función de "Port Forwarding".

- Para la comunicación de 1000 datos de tipo doble con un muestreo a 100 milisegundos sin cifrado o firmado digital se llega a tener un ancho de banda de 2.5 Mbps. Al disminuir el periodo de muestreo se observa un aumento proporcional de la cantidad de mensajes de respuesta de los valores de los datos y un aumento del ancho de banda. De igual forma se observa un aumento del ancho de banda al aumentar el número de datos a trasmitir. Al utilizar firmado digital se debe añadir 20 bytes a cada segmento de los valores transmitidos, con lo cual aumenta el tamaño del ancho de banda. En el siguiente cuadro se observa la estimación del ancho de banda para diferentes valores experimentados.

		Estimación del ancho de banda (Kbps)			
		Cantidad de datos de tipo doble			
		10	100	1000	10000
Tiempo de muestreo (ms)	100	62.28	286.92	2524.68	24923.88
	500	15.944	60.872	508.424	4988.264
	1000	10.152	32.616	256.392	2496.312

- El modelo de seguridad por medio de usuario y contraseña o por certificado al momento de establecer y autorizar el canal seguro se la realiza en el cliente OPC UA sin problemas al igual que la administración de los usuarios en el servidor OPC UA. El modelo es flexible y a la vez robusto, permitiendo configuraciones de seguridad a nivel de la red de planta, red de operaciones y red corporativa como se menciona en el marco teórico. Esto es importante porque permite escalar e interactuar a varios niveles. Se pudo observar adicionalmente la utilización de certificados X.509 v3 para la autenticación de aplicaciones a comunicar mediante el protocolo OPC UA, con la capacidad de denegar o autorizar aplicaciones al momento de comunicarse.

- De la captura de datos de las simulaciones se pudo observar que al tener una lectura de varios datos de tipo doble por separado consume mucha más longitud en bytes del mensaje OPC UA que al tener los datos juntos en un arreglo de varios datos de tipo doble. Al tener la lectura de datos separados de tipo doble, se observó que cada dato necesita de 30 bytes debido a que cada dato necesitaba de campos adicionales como tipo de datos y sello de tiempo, mientras al utilizar un arreglo de varios datos de tipo doble, cada miembro del arreglo únicamente utilizaba 8 bytes y únicamente el arreglo como objeto necesitaba de campos adicionales no cada dato. Por lo tanto la configuración que se recomendaría es de utilizar arreglos de datos de tipo doble.

- Al momento del análisis de las tramas OPC UA en el escenario de prueba de 1000 datos del tipo doble se puede observar la utilización de un tamaño de mensaje mayor al MTU, con lo cual se produce la segmentación del mensaje en varios pedazos de 1460 bytes que es el valor asignado por defecto. Este factor empieza a impactar el rendimiento de la red. Al momento de tener menos segmentación del

paquete se utiliza una menor cantidad de bytes debido al uso de cabeceras OPC UA y de firmas digitales. Por lo tanto se recomienda:

- Utilizar únicamente datos de tipo doble cuando se requiera valores con coma decimal de alta precisión, en su remplazo se pueden utilizar otro tipo de datos como Entero que son de menor tamaño en bytes pero de menor precisión.
- Utilizar varios datos de tipo doble en un arreglo de datos tipo doble con lo cual se tiene únicamente 8 bytes por cada miembro del arreglo de datos en lugar de 30 bytes.
- En lo posible utilizar tiempos de muestro rápidos para datos críticos y tiempos de muestreo lentos para datos no importantes.

- En la utilización de la comunicación OPC UA se puede observar que el descubrimiento de las variables por parte del software SCADA hacia los controladores industriales es inmediato sin configurar ningún permiso en el sistema operativo, aun en la lectura de arreglos de datos por medio de índices es de fácil implementación. El propósito de OPC es definir una interface común que se escribe una vez y volver a utilizar por cualquier sistema como SCADA, HMI, o paquetes de software personalizados. No hay nada en las especificaciones OPC para restringir el acceso al servidor a un dispositivo de control de procesos. Esto es interesante porque tradicionalmente cada vez que un software requería accesar a los datos de un dispositivo se debía escribir una interface (a veces llamados “drivers”). Con las pruebas se demuestra que la tecnología OPC UA es el remplazo efectivo para la tecnología OPC tradicional.

5.2 RECOMENDACIONES PARA FUTURAS INVESTIGACIONES.

- En el presente trabajo de investigación se tomó como base una simulación de lectura de datos del tipo doble, para futuras investigaciones se puede realizar casos de diferentes tipos de datos como por ejemplo en datos del tipo LREAL o

estructuras definidas de datos de monitoreo como por ejemplo en sistemas SCADA de Centrales Hidroeléctricas.

- El trabajo desarrollado utiliza un controlador industrial basado en PC el cual tiene un nivel de procesamiento alto y con comunicación sobre un tipo de red definida, para futuros trabajos de investigación se puede utilizar sistemas embebidos y con comunicación por medio de redes celulares o de radio para acceder a zonas remotas o de servicios públicos extendidos en una área grande.

- Para la toma de datos y el análisis de paquetes se utilizó el software de uso libre WireShark, si se requiere de un análisis más profundos se debería tomar en cuenta el uso de hardware y software dedicado para la captura de paquetes para el protocolo OPC UA.

5.3 BIBLIOGRAFÍA.

1. *Bechkoff. (2015, Noviembre). Twincat V3. Retrieved from http://infosys.beckhoff.com/index_en.htm*

2. *Iconics HMI/SCADA Software Developer. (2012). Genesis64 Standard Training Manual.*

3. *Mahnke, W., Leitner, S.-H., & Damm, M. (2009). OPC Unified Architecture. Berlin: Springer.*

4. *OPC-Foundation. (2015). Unified Architecture. Retrieved from https://opcfoundation.org/about/opc-technologies/opc-ua/*

5. *Rohjans, S., & Klaus, P. (2011). Standardized Smart Grid Semantics using OPC UA for Communication. IBIS - Interoperability in Business INformation Systems, 12.*

6. *Schwarz, M., & Borcsok, J. (2013). A Survey on OPC and OPC-UA. Communication and Automation Technologies, (p. 6). Sarajevo, Bosnia y Herzegovina.*